

Render without Edge Marks.



Render with Edge Marks enabled.

What are edge marks good for?

- When you need to render marks on an almost-flat plane, when other edge types cannot detect any line.
- When you want full control of edge rendering. Often used for edges of squarish shapes.
- Mark the whole base mesh to be rendered for base mesh preview.

What are edge marks not good for?

- Round outer edges (use instead *Contour/External Contour/Silhouette*).

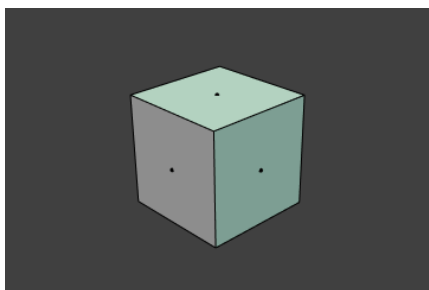
Face Marks

Face marks are useful for removing lines from certain areas of a mesh.

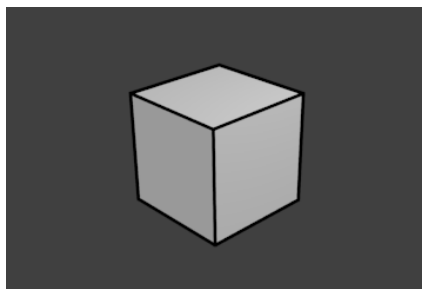
To set a face mark:

1. Select a mesh object and enter *Edit Mode*.
2. Select the faces you want to be marked.
3. Press **Ctrl-F** and select *Face Data ▶ Mark Freestyle Face*.

In this example, two faces of the default cube are marked like the image on the left. On the right is a render without face marks activated.



Marked faces (Edit Mode).



Render output.

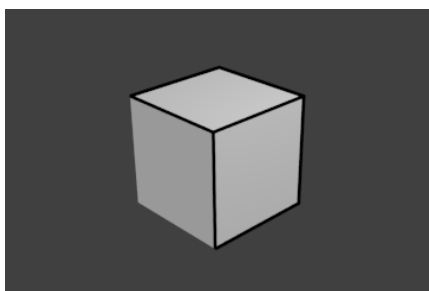
The line selection can be controlled via inclusion and faces options:

Negation

Whether to include or exclude edges matching defined face mark conditions from the line set.

Condition

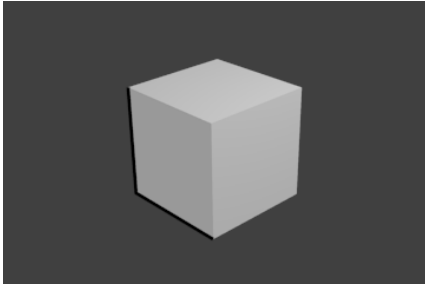
- One Face::** (De)select all edges which have one or both neighbor faces marked.
- Both Faces::** (De)select all edges which have both of their neighbor faces marked.



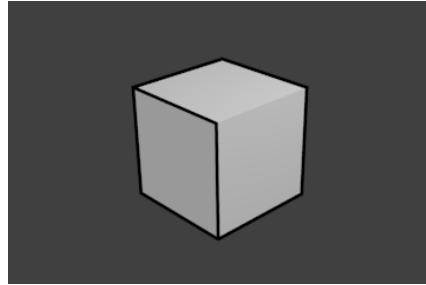
Inclusive, One Face.



Inclusive, Both Faces.



Exclusive, One Face.



Exclusive, Both Faces.

Collection

Include or exclude objects for line calculation, based on their belonging to a [Collection](#).

Line Set Collection

The name of the object collection to use.

Negation

Whether to include or exclude lines from those objects in this line set.

Line Style

- [Introduction](#)

Properties

- [Strokes](#)
- [Color](#)
- [Alpha](#)
- [Thickness](#)
- [Geometry](#)
- [Texture](#)

Modifiers

- [Color](#)
- [Alpha](#)
- [Thickness](#)
- [Geometry](#)

Introduction

In Freestyle, the line style settings define the appearance of a line set using five main aspects: Stroke, Color, Alpha, Thickness, Geometry, and Texture with each on a separate tab. These allow you to get many different styles of renders (technical draw, rough sketch, cartoon, calligraphy, etc.).

You can create as many line styles as you wish, and reuse a given line style for several line sets by selecting it from the select menu next to its name.

Note

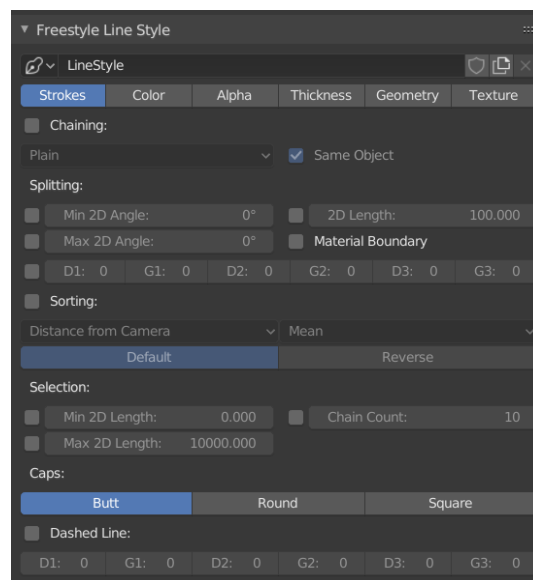
Unless otherwise specified, all lengths in line style settings are in pixels (either relative or absolute, as specified in the [core options](#)).



Line Style Example ([blend-file](#)).

Strokes

Strokes are the final rendered lines. Yet you can tweak them, for example, by removing the ones longer/shorter than some threshold, chaining lines into a single stroke or breaking a stroke into several ones based on angles, dashed pattern, etc.

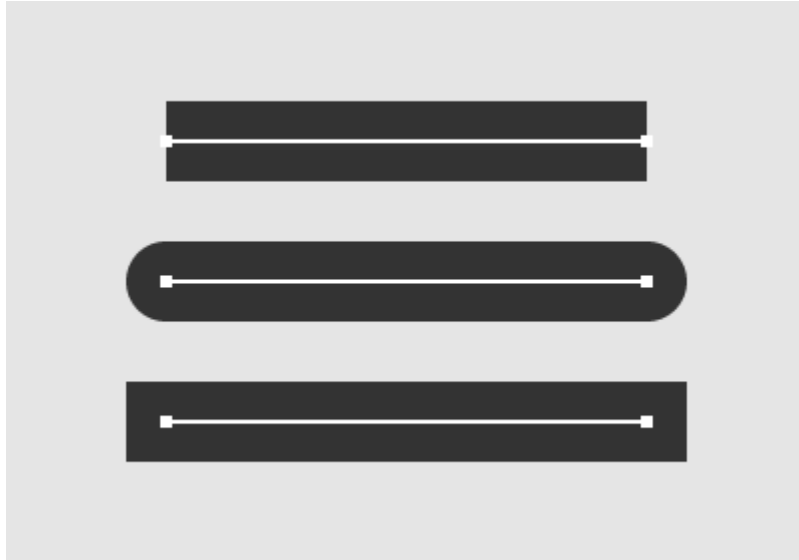


Line Style: Strokes.

Caps

You can choose between three types of line caps:

- | | |
|-----------------|---|
| Butt:: | Flat cap, exactly at the point the line ends. |
| Round:: | A half circle centered on the end point of the line. |
| Square:: | A square centered on the end point of the line (hence, like the circle, the drawn end of the line is slightly extended compared to its computed value). |



Line caps example.

Chaining

By default all retrieved lines from the line set are chained together. There are two basic chaining methods:

Method

Plain::

The default chaining method; it creates simple chains.

Sketchy::

This chaining option allows for generating chains of feature edges with sketchy multiple strokes. Basically, it generates *Round* strokes instead of a single one. It is only really useful if you use some random-driven modifiers in the line style!

Rounds

It specifies the number of rounds in sketchy strokes.

Same Object

If true, only feature edges of the same object are joined.

Chaining can also be turned off to render each line separately, which can be useful for line styles which depend on accurate representation of the line

set.

Splitting

You can split up chains of Freestyle lines by enabling one of the following:

Min/Max 2D Angle

Splits chains of feature edges when they make a 2D angle above (or below) a minimum (or maximum) threshold.

2D Length

Splits chains when they are longer than the given value.

Material Boundary

Splits chains of feature edges if they cross from one material to another.

Split Pattern

Splits the chains using the given dashed pattern (see also [Dashed Line](#)).

Dash 1, 2, 3

Length of the specified dash for splitting.

Gap 1, 2, 3

Length of the specified gap for splitting.

Sorting

You can sort the order of your strokes, allowing the lines to stack in the order given.

Sort Key

A sort key is used to determine the stacking order of lines.

Distance from Camera::

Lines closer to the camera lie on top of further lines.

2D Length::	Longer lines lie on top of shorter lines.
Projected X/Y::	Sort by the projected X or Y value in the image coordinate system.

Integration Type

Use in tandem with the Sort Key to determine the range for sorting. Since the distance of a line from the camera may vary over vertices, this option computes the sort key for a line from the values computed at individual vertices. The value computed for the line is:

Mean::	The mean of the values obtained for the vertices.
Min::	The minimum of the values obtained for the vertices.
Max::	The maximum of the values obtained for the vertices.
First::	The value obtained for the first vertex.
Last::	The value obtained for the last vertex.

Sort Order

With the given result you can choose to “Reverse” the sort order.

Selection

You can also choose to only render selected chains.

Min/Max 2D Length

Chains longer and/or shorter than *2D Length*.

Chain Count

Allows the selection of first N chains.

Dashed Line

By enabling the *Dashed Line* checkbox, you can specify three pairs of dash and gap lengths. Dash values define the lengths of dash strokes, while gap values specify intervals between two dashes.

If a zero gap is specified, then the corresponding dash is ignored even if it has a nonzero value.

Dashes are treated as separate strokes, meaning that you can apply line caps, as well as color, alpha and thickness modifiers.

Dash 1, 2, 3

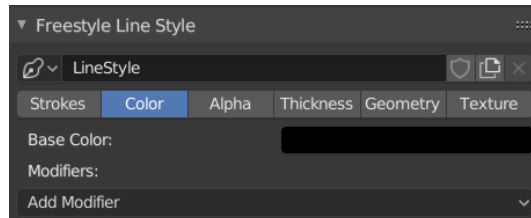
Length of the specified dash for dashed lines.

Gap 1, 2, 3

Length of the specified gap for dashed lines.

Color

In this tab you control the color of your strokes.



Line Style: Color.

Base Color

The base color for this line style.

Modifiers

Common Options

Mix

The modifier output can be mixed with the base property using the usual methods (see for example the [Mix compositing node](#)).

Influence

How much the result of this modifier affects the current property.

Color Ramp

Each modifier has [color ramp](#) that maps the property to a stroke color.

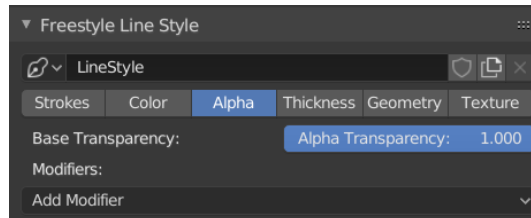
Types

- [Along Stroke](#)
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)

- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Alpha

In this tab you control the alpha (transparency) of your strokes.



Line Style: Alpha.

Base Transparency

The base alpha for this line style.

Modifiers

Common Options

Mix

The modifier output can be mixed with the base property using the usual methods (see for example the [Mix compositing node](#)).

Influence

How much the result of this modifier affects the current property.

Mapping

Either a linear progression (from 0.0 to 1.0), or a custom mapping [curve](#).

Note

Note the linear non-inverted option is equivalent to “do nothing”, as original values from materials are already in the (0.0 to 1.0) range.

That is the case for: Crease Angle, Curvature 3D, Material, Noise, Tangent.

Invert

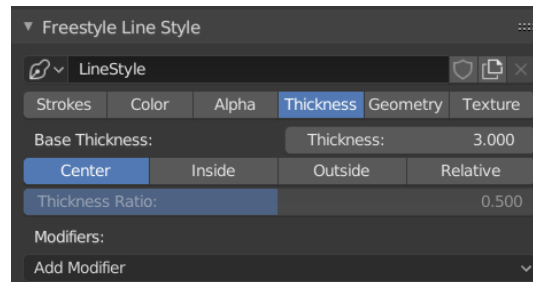
Inverts the *Mapping*.

Types

- [Along Stroke](#)
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)
- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Thickness

Controls the thickness of the Freestyle strokes.



Line Style: Thickness.

Base Thickness

The base thickness for this line style.

Thickness Position

Control the position of stroke thickness from the original (backbone) stroke geometry. There are four choices:

- | | |
|-------------------|--|
| Center:: | The thickness is evenly split to the left and right side of the stroke geometry. |
| Inside:: | The strokes are drawn within object boundary. |
| Outside:: | The strokes are drawn outside the object boundary. |
| Relative:: | Specifies the relative position by a number between 0.0 (inside) and 1.0 (outside), in the <i>Thickness Ratio</i> number field just below. |

Note

The thickness position options are applied only to strokes of edge types *Silhouette* and *Border*, since these are the only edge types

defined in terms of the object boundary. Strokes of other edge types are always drawn using the *Center* option.

Modifiers

Common Options

Mix

The modifier output can be mixed with the base property using the usual methods (see for example the [Mix compositing node](#)).

Influence

How much the result of this modifier affects the current property.

Types

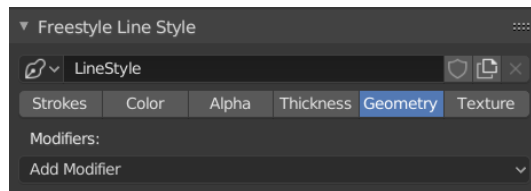
- [Along Stroke](#)
- [Calligraphy](#)
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)
- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Geometry

In this tab you control the geometry of your strokes. It contains only the option to add modifiers.

As they always completely apply to the strokes' geometry (like object modifiers do). They take the resulting two-dimensional strokes from the Freestyle line set and displace or deform them in various ways.

As with other modifier stacks in Blender, they are applied from top to bottom.



Line Style: Geometry.

Modifiers

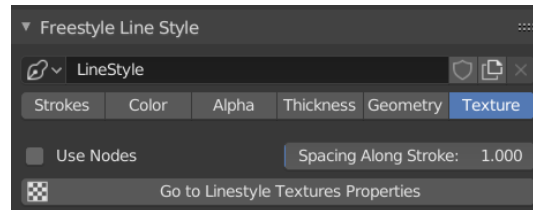
Types

- [2D Offset](#)
- [2D Transform](#)
- [Backbone Stretcher](#)
- [Bézier Curve](#)
- [Blueprint](#)
- [Guiding Lines](#)
- [Perlin Noise 1D](#)
- [Perlin Noise 2D](#)
- [Polygonization](#)
- [Sampling](#)
- [Simplification](#)

- [Sinus Displacement](#)
- [Spatial Noise](#)
- [Tip Remover](#)

Texture

Assigns a texture to the Freestyle stroke.



Line Style: Texture.

Use Nodes

In Cycles textures are defined by means of shader [Nodes](#).

Spacing Along Stroke

Allows to set the “pace” of textures mapped along the length of strokes.

Go to Linestyle Textures

A shortcut to the line style texture properties in the other *Textures* tab of the Properties. Make sure to first enable *Use Nodes*.

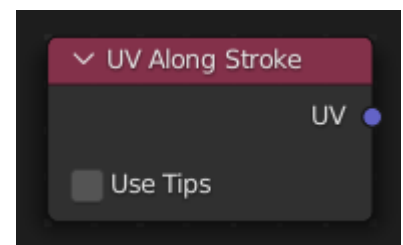
Nodes

UV Along Stroke Node

The *UV Along Stroke* input node is maps textures along the stroke length, making it possible to mimic pencil, paintbrush, and other art medium marks.

Note

These UV maps become available only during the Freestyle rendering process. Hence, the UV



UV Along Stroke Node.

Along Stroke node cannot be replaced by the conventional UV Map input node which takes an existing UV map already defined as part of mesh data.

Inputs

This node has no inputs.

Properties

Use Tips

Allows to use lower quarters of a texture image for the head and tail tips of a stroke, while the upper half for the stroke body.

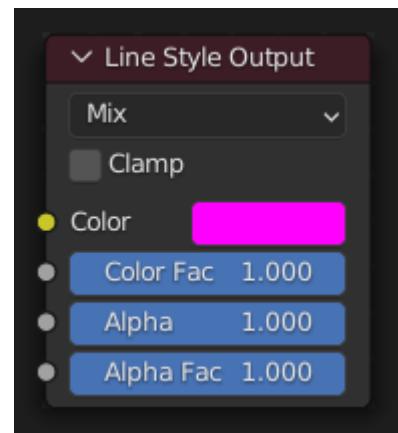
Outputs

UV

UV maps defined along strokes.

Line Style Output Node

The *Line Style Output* node specifies how to mix the texture information into the base color of line styles.



Line Style Output Node.

Inputs

Color

Color input for the texture.

Color Factor

Standard mix factor of the *Color* value.

Alpha

Alpha input for the texture.

Alpha Factor

Standard mix factor of the *Alpha* value.

Properties

Mix

The Blend mode can be selected in the select menu. See [Color Blend Modes](#) for details on each blending mode.

Clamp

Limit the highest color value to not exceed 1.0.

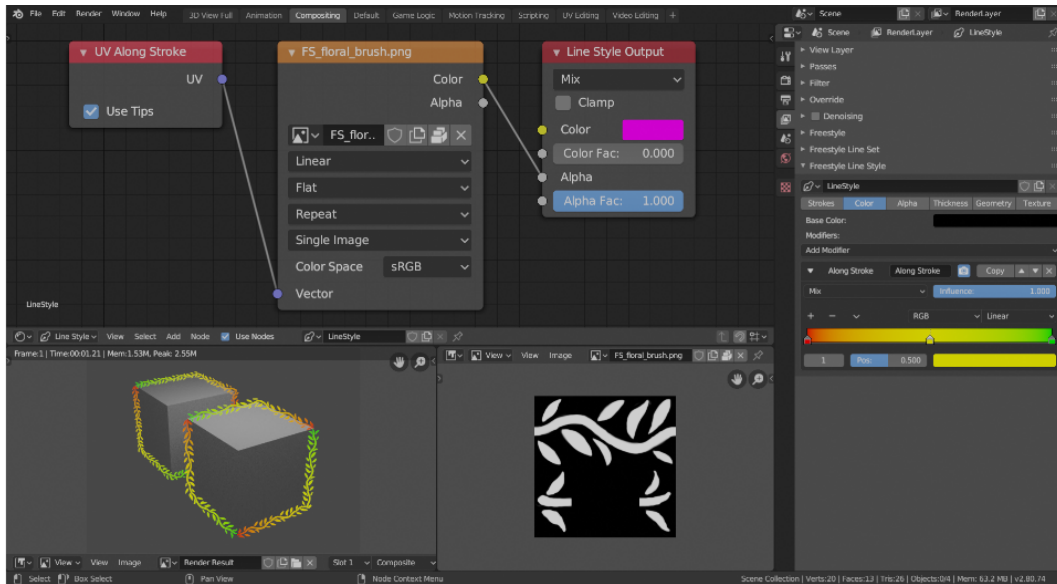
Outputs

This node has no outputs.

Example

The image below shows a typical shader node tree that maps a floral texture image along strokes. The UV Along Stroke input node retrieves UV maps defined by Freestyle along generated strokes, and passes them to the Vector input channel of the Image Texture node. A texture image is selected in the Image Texture node, and its color is inputted to the Alpha channel of the Line Style Output node. Since the Alpha Factor is set to one, the texture image replaces the base alpha transparency of the active line style (shown

in the Freestyle Line Style panel). On the other hand, the Mix blend mode is selected in the Line Style Output node with the Color Factor set to zero, so that the gradient line color specified in the active line style is applied along strokes.



Example of Line Style Nodes ([blend-file](#)).

It is noted that the texture image `FS_floral_brush.png` shown in the screen capture is an example of Freestyle brush images with tips. Specifically, the upper half of the image is used as a seamless horizontal tile of the stroke body. Whereas the parts in the lower half are tips (stroke caps) at both ends of the stroke.

Color Modifiers

- [Along Stroke](#)
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)
- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Along Stroke

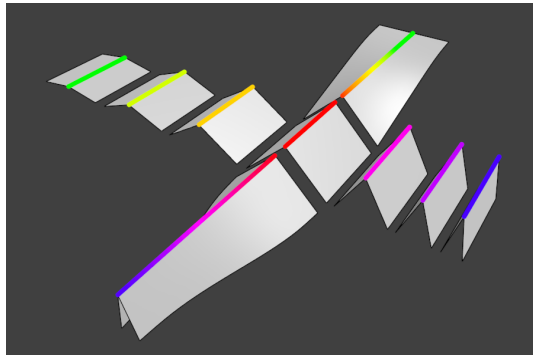
The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max

The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



Crease Angle modifier example by T.K. ([blend-file](#)).

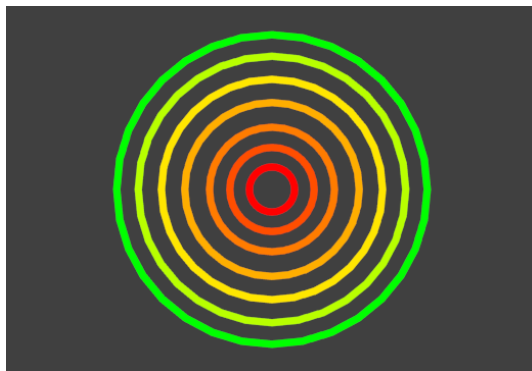
Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The [curvature](#) of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max

The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. ([blend-file](#)).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target

The object to measure distance from.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. ([blend-file](#)).

Noise

The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude

The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period

The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

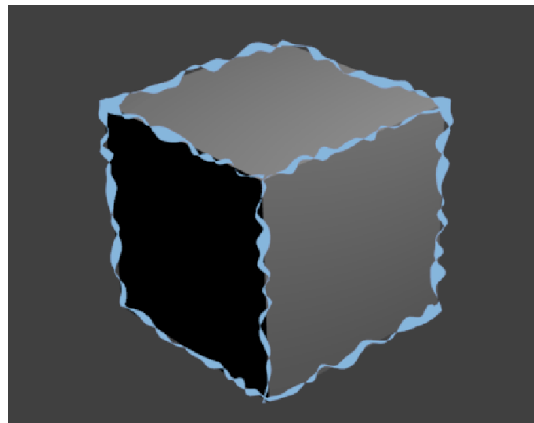
Seed

Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*

Allows the thickness to be distributed unevenly at every point.

Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using
asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.

Alpha Modifiers

- [Along Stroke](#)
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)
- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Along Stroke

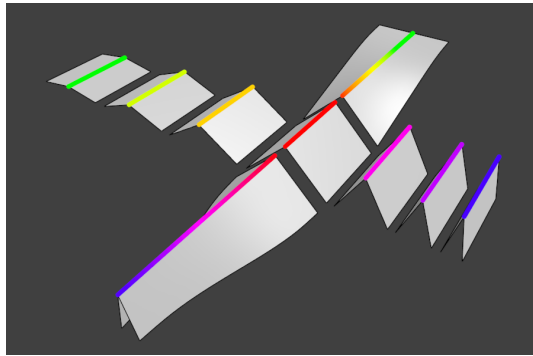
The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max

The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



Crease Angle modifier example by T.K. ([blend-file](#)).

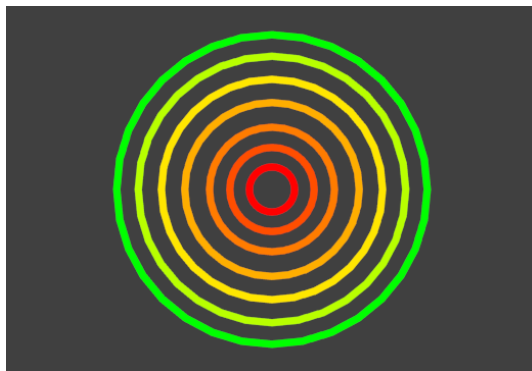
Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The [curvature](#) of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max

The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. ([blend-file](#)).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target

The object to measure distance from.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. ([blend-file](#)).

Noise

The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude

The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period

The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

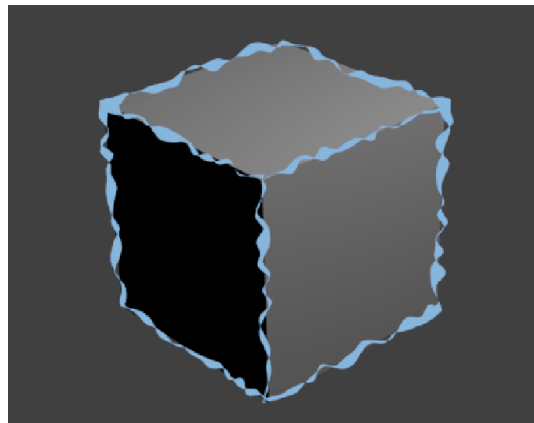
Seed

Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*

Allows the thickness to be distributed unevenly at every point.

Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using
asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.

Thickness Modifiers

- [Along Stroke](#)
- [Calligraphy](#).
- [Crease Angle](#)
- [Curvature 3D](#)
- [Distance from Camera](#)
- [Distance from Object](#)
- [Material](#)
- [Noise](#)
- [Tangent](#)

Along Stroke

The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Calligraphy

The *Calligraphy* modifier mimics some broad and flat pens for calligraphy. It generates different thickness based on the orientation of the stroke.

Orientation

The angle (orientation) of the virtual drawing tool, from the vertical axis of the picture. For example, an angle of 0.0 mimics a pen aligned with the vertical axis. Hence, the thickest strokes will be the vertical ones i.e. stroke's direction is aligned with the angle, and the thinnest will be the horizontal ones i.e. stroke's direction is perpendicular to the angle.



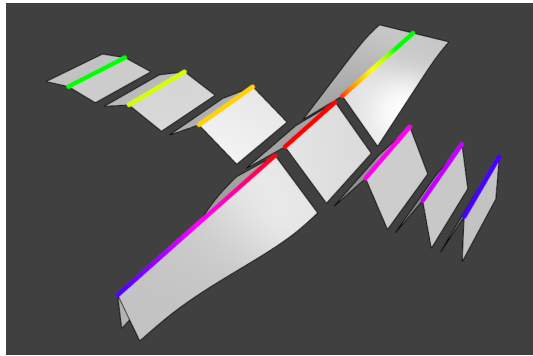
Calligraphy modifier demo by T.K. ([blend-file](#)).

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max

The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



Crease Angle modifier example by T.K. ([blend-file](#)).

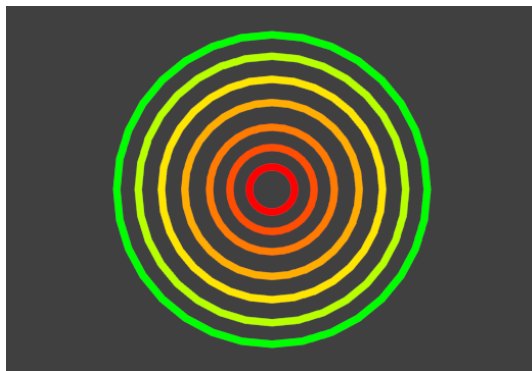
Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The [curvature](#) of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max

The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. ([blend-file](#)).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target

The object to measure distance from.

Range Min, Max

The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection

Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. ([blend-file](#)).

Noise

The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude

The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period

The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

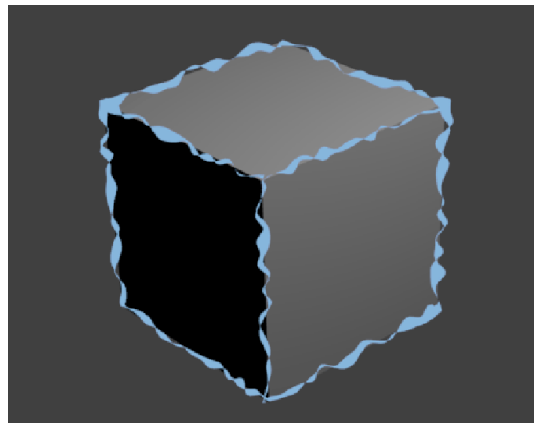
Seed

Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*

Allows the thickness to be distributed unevenly at every point.

Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using
asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke's vertices.

Geometry Modifiers

- [2D Offset](#)
- [2D Transform](#)
- [Backbone Stretcher](#)
- [Bézier Curve](#)
- [Blueprint](#)
- [Guiding Lines](#)
- [Perlin Noise 1D](#)
- [Perlin Noise 2D](#)
- [Polygonization](#)
- [Sampling](#)
- [Simplification](#)
- [Sinus Displacement](#)
- [Spatial Noise](#)
- [Tip Remover](#)

2D Offset

The *2D Offset* modifier adds some two-dimensional offsets to the stroke backbone geometry. It has two sets of independent options/effects:

Start, End

These two options add the given amount of offset to the start (or end) point of the stroke, along the (2D) normal at those points. The effect is blended over the whole stroke, if you for example, set only *Start* to 50, the start of the stroke is offset 50 pixels along its normal, the middle of the stroke, 25 pixels along its own normal, and the end point is not moved.

X, Y

These two options simply add a constant horizontal and/or vertical offset to the whole stroke.

2D Transform

The *2D Transform* modifier applies two-dimensional scaling and/or rotation to the stroke backbone geometry. Scale is applied before rotation.

Pivot

The center (pivot point) of these 2D transformations can be:

Stroke Center:: The median point of the stroke.

Stroke Start:: The beginning point of the stroke.

Stroke End:: The end point of the stroke.

Stroke Point Parameter::

The *Stroke Point Parameter* factor controls where along the stroke the pivot point is (start point if set to 0.0; end point if set to 1.0).

Absolute 2D Point:: The *Pivot X* and *Y* values define the position of the pivot point in the final render (from the bottom left corner).

Important

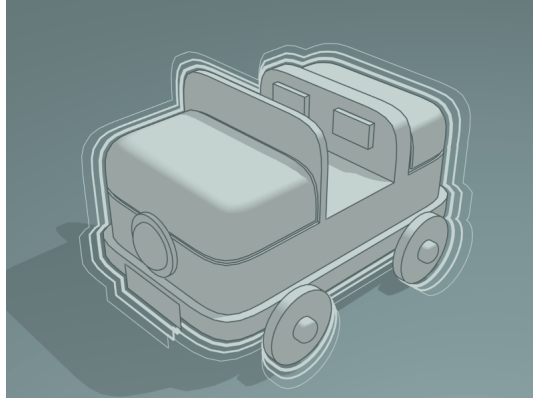
Currently, you have to take into account the *real* render size, i.e. resolution **and** resolution percentage.

Scale X, Y

The scaling factors, in their respective axes.

Rotation Angle

The rotation angle.



2D Transform modifier ([blend-file](#)).

Backbone Stretcher

The *Backbone Stretcher* modifier stretches (adds some length to) the beginning and end of the stroke.

Backbone Length

Length to add to the strokes' ends.

Bézier Curve

The *Bézier Curve* modifier replaces the stroke by a Bézier approximation of it.

Error

The maximum distance allowed between the new Bézier curve and the original stroke.



Bézier Curve modifier demo by T.K. ([blend-file](#)).

Blueprint

The *Blueprint* modifier produces blueprint-like strokes using either circular, elliptical, or square contours. A blueprint here refers to those lines drawn at the beginning of free-hand drawing to capture the silhouette of objects with a simple shape such as circles, ellipses and squares.

Shape

Which base shapes to use for this blueprint: *Circles*, *Ellipses* or *Squares*.

Rounds

How many rounds are generated, as if the pen draws the same stroke several times (i.e. how many times the process is repeated).

Random Radius, Center

For the *Circles* and *Ellipses* shapes. Adds some randomness to each round in the relevant aspect. Using more than one round with no randomness would be meaningless, as they would draw over each other exactly.

Backbone Length, Backbone

For the *Squares* shapes. The first adds some extra length to each edge of the generated squares (also affected by the second parameter). The second adds some randomness to the squares.

Note that the *Min 2D Length* feature from the *Strokes* settings is quite handy here, to avoid the noise generated by small strokes...

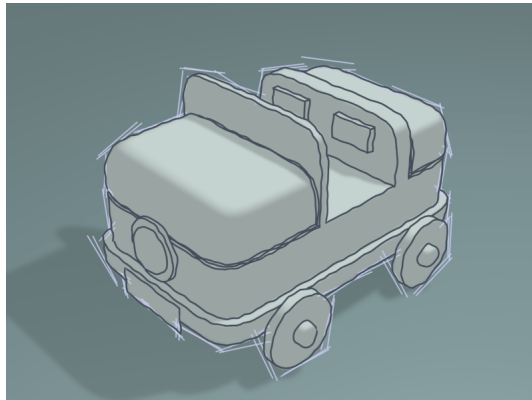
Guiding Lines

The *Guiding Lines* modifier replaces a stroke by a straight line connecting both of its ends.

Offset

Offset the start and end points along the original stroke, before generating the new straight one.

This modifier will produce reasonable results when strokes are short enough, because shorter strokes are more likely to be well approximated by straight lines. Therefore, it is recommended to use this modifier together with one of the splitting options (by 2D angle or by 2D length) from the *Strokes* panel.



Guiding Lines modifier Demo by T.K. ([blend-file](#)).

Perlin Noise 1D

The *Perlin Noise 1D* modifier adds one-dimensional Perlin noise to the stroke. The curvilinear abscissa (value between 0 and 1 determined by a point's position relative to the first and last point of a stroke) is used as the input to the noise function to generate noisy displacements.

This means that this modifier will give an identical result for two strokes with the same length and sampling interval.

Frequency

How dense the noise is (kind of a scale factor along the stroke).

Amplitude

How much the noise distorts the stroke in the *Angle* direction.

Seed

The seed of the random generator (the same seed over a stroke will always give the same result).

Octaves

The “level of detail” of the noise.

Angle

In which direction the noise is applied (0.0 is fully horizontal).

Perlin Noise 2D

The *Perlin Noise 2D* modifier adds one-dimensional Perlin noise to the stroke. The modifier generates noisy displacements using 2D coordinates of stroke vertices as the input of the noise generator.

Frequency

How dense the noise is (kind of a scale factor along the stroke).

Amplitude

How much the noise distorts the stroke in the *Angle* direction.

Seed

The seed of the random generator (the same seed over a stroke will always give the same result).

Octaves

The “level of detail” of the noise.

Angle

In which direction the noise is applied (0.0 is fully horizontal).

Polygonization

The *Polygonization* modifier simplifies strokes as much as possible (in other words, it transforms smooth strokes into jagged polylines).

Error

The maximum distance allowed between the new simplified stroke and the original one (the larger this value is, the more jagged/approximated the resulting polylines are).

Sampling

The *Sampling* modifier changes the definition, precision of the stroke, for the following modifiers.

Sampling

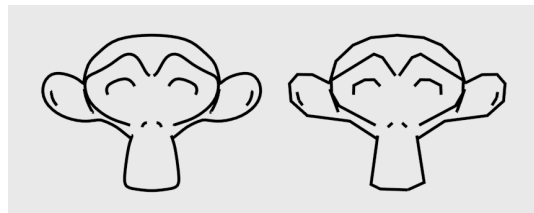
The smaller this value, the more precise are the strokes. Be careful; too small values will require a huge amount of time and memory during render!

Simplification

The *Simplification* modifier merges stroke vertices that lie close to one another, like the *Decimate* modifier for meshes.

Tolerance

Measure for how close points have to be to each other to be merged. A higher tolerance means more vertices are merged.



Sinus Displacement

The *Sinus Displacement* modifier adds a sinusoidal displacement to the stroke.

Wavelength

How wide the undulations are along the stroke.

Amplitude

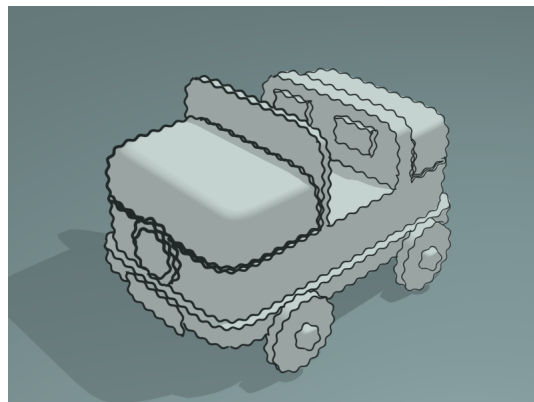
How high the undulations are across the stroke.

Phase

Allows “offsetting” (“moving”) the undulations along the stroke.

Tip

The undulations this modifier produces look exactly the same at a Phase of 0 and any positive or negative multiple of the Wavelength set on the modifier. This can be used for rendering short video sequences with wavy lines that can then be seamlessly looped without any visual jumps in the undulations along the line.



Sinus Displacement modifier demo by T.K. ([blend-file](#)).

Spatial Noise

The *Spatial Noise* modifier adds some spatial noise to the stroke. Spatial noise displacements are added in the normal direction (i.e. the direction perpendicular to the tangent line) evaluated at each stroke vertex.

Amplitude

How much the noise distorts the stroke.

Scale

How wide the noise is along the stroke.

Octaves

The level of detail of the noise.

Smooth

When enabled, apply some smoothing over the generated noise.

Pure Random

When disabled, the next generated random value depends on the previous one; otherwise they are completely independent. Disabling this setting gives a more “consistent” noise along a stroke.

Tip Remover

The *Tip Remover* modifier removes a piece of the stroke at its beginning and end.

Tip Length

Length of stroke to remove at both of its tips.

Material Properties

Reference

Panel:: *Properties ▶ Material ▶ Freestyle Line*

Line Color

Specifies the line colors on a per-material basis.

Priority

Specify the ordering of competing line colors at material boundaries.

See also

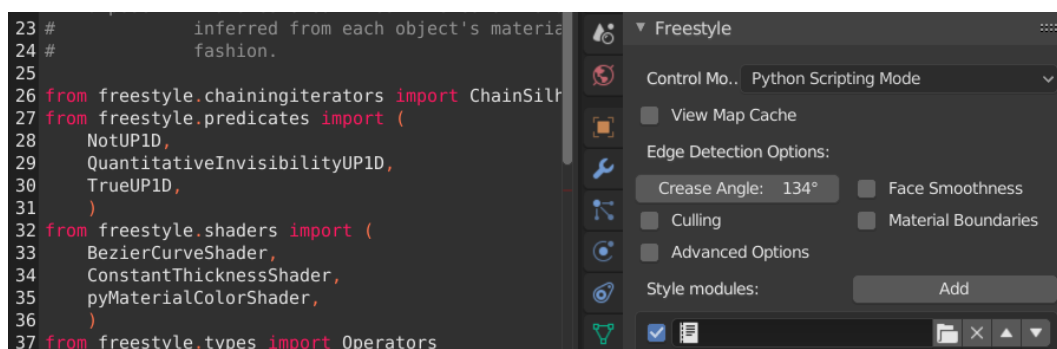
A use case of the line color priority is detailed in a Freestyle development [blog article](#).

Python Scripting

The Python Scripting mode offers full programmable line stylizes. In this control mode, all styling operations are written as Python scripts referred to as style modules in the Freestyle terminology. The input to a style module is a view map (i.e. a set of detected feature edges), and the output is a set of stylized strokes.

A style module is composed of successive calls of five basic operators: selection, chaining, splitting, sorting and stroke creation. The selection operator identifies a subset of input feature edges based on one or more user-defined selection conditions (predicates). The selected edges are processed with the chaining, splitting and sorting operators to build chains of feature edges. These operators are also controlled by user-supplied predicates and functions in order to determine how to transform the feature edges into chains. Finally, the chains are transformed into stylized strokes by the stroke creation operator, which takes a list of user-defined stroke shaders.

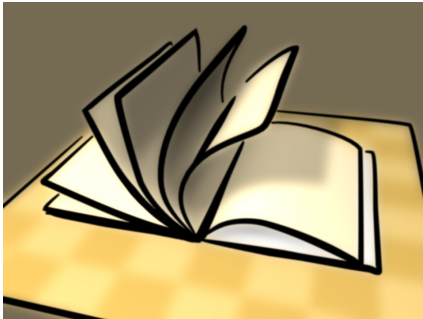
Python style modules are stored within blend-files as text data-blocks. External style module files first need to be loaded in the Text Editor. Then the select menu within an entry of the style module stack allows you to select a module from the list of loaded style modules.



A screen capture of a style module `cartoon.py` loaded in the Text Editor (left), as well as Freestyle options in the Python Scripting

mode in the View Layers buttons (right).

Freestyle for Blender comes with a number of Python style modules that can serve as a starting point of your own style module writing. See also the section of the Freestyle Python API in the Blender Python API reference manual for the full detail of style module constructs.



By T.K. using the Python Scripting mode ([blend-file](#), CC0).



By T.K. using the Python Scripting mode ([blend-file](#), CC0).

Writing Style Modules

A style module is a piece of code responsible for the stylization of Freestyle line drawing. The input of a style module is a set of feature edges called view map (ViewMap). The output is a set of stylized lines also referred to as strokes. A style module is structured as a pipeline of operations that allow for building strokes from the input edges within the view map.

There are five kinds of operations (listed with corresponding operator functions):

- Selection `Operators.select()`
- Chaining `Operators.chain()`, `Operators.bidirectional_chain()`
- Splitting `Operators.sequential_split()`, `Operators.recursive_split()`
- Sorting `Operators.sort()`

- Stroke creation `Operators.create()`

The input view map is populated with a set of `ViewEdge` objects. The selection operation is used to pick up `ViewEdges` of interest to artists based on user-defined selection conditions (predicates). Chaining operations take the subset of `ViewEdges` and build `Chains` by concatenating `ViewEdges` according to user-defined predicates and functions. The `Chains` can be further refined by splitting them into smaller pieces (e.g. at points where edges make an acute turn) and selecting a fraction of them (e.g. to keep only those longer than a length threshold). The sorting operation is used to arrange the stacking order of chains to draw one line on top of another. The chains are finally transformed into stylized strokes by the stroke creation operation applying a series of stroke shaders to individual chains.

`ViewEdges`, `Chains` and `Strokes` are generically referred to as one-dimensional (1D) elements. A 1D element is a polyline that is a series of connected straight lines. Vertices of 1D elements are called 0D elements in general.

All the operators act on a set of active 1D elements. The initial active set is the set of `ViewEdges` in the input view map. The active set is updated by the operators.

Selection

The selection operator goes through every element of the active set and keeps only the ones satisfying a certain predicate. The `Operators.select()` method takes as the argument a unary predicate that works on any `Interface1D` that represents a 1D element. For example:

```
Operators.select(QuantitativeInvisibilityUP1D(0))
```

This selection operation uses the `QuantitativeInvisibilityUP1D` predicate to select only the visible `ViewEdge` (more precisely, those whose quantitative invisibility is equal to 0). The selection operator is intended to selectively apply the style to a fraction of the active 1D elements.

It is noted that `QuantitativeInvisibilityUP1D` is a class implementing the predicate that tests line visibility, and the `Operators.select()` method takes an instance of the predicate class as argument. The testing of the predicate for a given 1D element is actually done by calling the predicate instance, that is, by invoking the `__call__` method of the predicate class. In other words, the `Operators.select()` method takes as argument a functor which in turn takes an `Interface0D` object as argument. The Freestyle Python API employs functors extensively to implement predicates, as well as functions.

Chaining

The chaining operators act on the set of active `ViewEdge` objects and determine the topology of the future strokes. The idea is to implement an iterator to traverse the `ViewMap` graph by marching along `ViewEdges`. The iterator defines a chaining rule that determines the next `ViewEdge` to follow at a given vertex (see `ViewEdgeIterator`). Several such iterators are provided as part of the Freestyle Python API (see `ChainPredicateIterator` and `ChainSilhouetteIterator`). Custom iterators can be defined by inheriting the `ViewEdgeIterator` class. The chaining operator also takes as argument a `UnaryPredicate` working on `Interface1D` as a stopping criteria. The chaining stops when the iterator has reached a `ViewEdge` satisfying this predicate during the march along the graph.

Chaining can be either unidirectional `Operators.chain()` or bidirectional `Operators.bidirectional_chain()`. In the latter case, the chaining will propagate in the two directions from the starting edge.

The following is a code example of bidirectional chaining:

```
Operators.bidirectional_chain(  
    ChainSilhouetteIterator(),  
    NotUP1D(QuantitativeInvisibilityUP1D(0)),  
)
```

The chaining operator uses the `ChainSilhouetteIterator` as the chaining rule and stops chaining as soon as the iterator has come to an invisible `ViewEdge`.

The chaining operators process the set of active `ViewEdge` objects in order. The active `ViewEdges` can be previously sorted using the `Operators.sort()` method (see below). It starts a chain with the first `ViewEdge` of the active set. All `ViewEdges` that have already been involved in the chaining process are marked (in the case of the example above, the time stamp of each `ViewEdge` is modified by default), in order not to process the same `ViewEdge` twice. Once the chaining reaches a `ViewEdge` that satisfies the stopping predicate, the chain is terminated. Then a new chain is started from the first unmarked `ViewEdge` in the active set. This operation is repeated until the last unmarked `ViewEdge` of the active set was processed. At the end of the chaining operation, the active set is set to the Chains that have just been constructed.

Splitting

The splitting operation is used to refine the topology of each Chain. Splitting is performed either sequentially or recursively. Sequential splitting `Operators.sequentialSplit()` in its basic form, parses the Chain at a given arbitrary resolution and evaluates a unary predicate (working on 0D elements) at each point along the Chain. Every time the predicate is satisfied, the chain is split into two chains. At the end of the sequential split operation, the active set of chains is set to the new chains.

```
Operators.sequentialSplit(TrueUP0D(), 2)
```

In this example, the chain is split every 2 units. A more elaborated version uses two predicates instead of one: One to determine the starting point of the new chain and the other to determine its ending point. This second version can lead to a set of Chains that are disjoint or that overlap if the two predicates are different (see `Operators.sequentialSplit()` for more details).

Recursive splitting `Operators.recursiveSplit()` evaluates a function on the 0D elements along the Chain at a given resolution and find the point that gives the maximum value for the function. The Chain is then split into two at that point. This process is recursively repeated on each of the two new Chains, until the input Chain satisfies a user-specified stopping condition.

```
func = Curvature2DAngleF0D()  
Operators.recursive_split(func, NotUP1D(HigherLengthUP1D(5)),  
5)
```

In the code example above, the Chains are recursively split at points of the highest 2D curvature. The curvature is evaluated at points along the Chain at a resolution of 5 units. Chains shorter than 5 units will not be split anymore.

Sorting

The sorting operator `Operators.sort()` arranges the stacking order of active 1D elements. It takes as argument a binary predicate used as a “smaller than” operator to order two 1D elements.

```
Operators.sort(Length2DBP1D())
```

In this code example, the sorting uses the `Length2DBP1D` binary predicate to sort the `Interface1D` objects in the ascending order in terms of 2D length.

The sorting is particularly useful when combined with causal density. Indeed, the causal density evaluates the density of the resulting image as it is modified. If we wish to use such a tool to decide to remove strokes whenever the local density is too high, it is important to control the order in which the strokes are drawn. In this case, we would use the sorting operator to ensure that the most “important” lines are drawn first.

Stroke Creation

Finally, the stroke creation operator `Operators.create()` takes the active set of Chains as input and build Strokes. The operator takes two arguments. The first is a unary predicate that works on `Interface1D` that is designed to make a last selection on the set of chains. A Chain that does not satisfy the condition will not lead to a Stroke. The second input is a list of shaders that will be responsible for the shading of each built stroke.

```
shaders_list = [  
    SamplingShader(5.0),
```

```
        ConstantThicknessShader(2),  
        ConstantColorShader(0.2,0.2,0.2,1),  
    ]  
Operators.create(DensityUP1D(8,0.1, IntegrationType.MEAN),  
shaders_list)
```

In this example, the `DensityUP1D` predicate is used to remove all Chains whose mean density is higher than 0.1. Each chain is transformed into a stroke by resampling it so as to have a point every 5 units and assigning to it a constant thickness of 2 units and a dark gray constant color.

User Control on the Pipeline Definition

Style module writing offers different types of user control, even though individual style modules have a fixed pipeline structure. One is the sequencing of different pipeline control structures, and another is through the definition of functor objects that are passed as argument all along the pipeline.

Different pipeline control structures can be defined by sequencing the selection, chaining, splitting, and sorting operations. The stroke creation is always the last operation that concludes a style module.

Predicates, functions, chaining iterators, and stroke shaders can be defined by inheriting base classes and overriding appropriate methods. See the reference manual entries of the following base classes for more information on the user-scriptable constructs.

See also

Predicates, functions, chaining iterators, and stroke shaders can be defined by inheriting base classes and overriding appropriate methods. See [Freestyle python module](#) for more information on the user-scriptable constructs.

Layers & Passes

- [Introduction](#)
 - [View Layers](#)
 - [Usage](#)
 - [Collections](#)
 - [Cycles](#)
- [View Layer](#)
- [Passes](#)
 - [Data](#)
 - [Light](#)
 - [Cryptomatte](#)
 - [Shader AOV](#)
 - [Light Groups](#)
 - [Combining](#)
 - [Known Limitations](#)

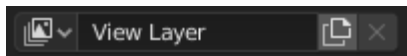
Introduction

Renders can be separated into layers, to composite them back together afterwards.

Some example usages are applying compositing effects to characters separately, blurring the background and foreground layers separately for depth of field, or rendering different lighting variations of the same scene.

Using View Layers can also save you from having to re-render your entire image after each change, allowing you to instead re-render only the layer(s) that you have altered.

View Layers



View Layers.

In the top of the screen there is a list of all the View Layers in the active scene.

Name

The name of the active view layer, click to edit the name.

Add View Layer

Will add a new view layer to the active scene.

New

Adds a new view layer.

Copy Settings

Adds a new view layer with all the settings of current view layer.

Blank

Adds a new view layer with all collections disabled.

Remove View Layer

Will remove the selected view layer from the active scene.

Note

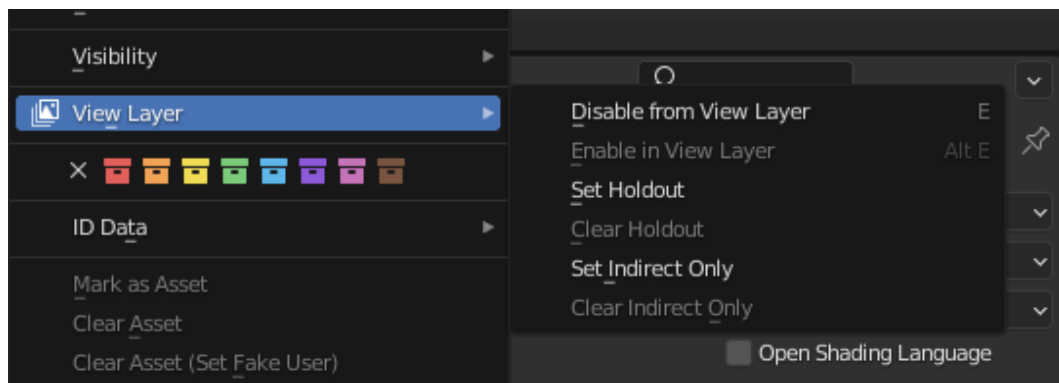
A scene must have at least one view layer.

Usage

Each [Scene](#) has an associated set of [Collections](#). The visibility settings of each collection can be changed per View Layer to separate the rendering of different objects and lights into layers.

Collections

Per collection you can adjust the way how the render engine needs to render the objects inside. Based on the render engine different options can be set.



Collection/View layer settings.

Disable from View Layer

Remove this collection from the active view layer. Objects that are only in this collection will not be rendered for the active view layer. This is

useful to sometimes leave out some object influence for a particular view layer.

Enable in View Layer

Add this collection to the active view layer. Objects inside the collection will be rendered with the active view layer.

Set Holdout

Objects inside this collection will generate a holdout/mask in the active view layer.

Clear Holdout

Clear the Set Holdout flag.

Set Indirect Only *Cycles Only*

Objects inside this collection will only contribute to the final image indirectly through shadows and reflections.

Clear Indirect Only *Cycles Only*

Clear the *Indirect Only* flag. Objects inside this collection will contribute normally to the final image.

Cycles

Reference

Panel:: *View Layers ▸ Layer*

This section covers only the Render Layer settings appropriate for the Cycles renderer. For the engine-independent settings, see [this section](#).

Filter

Include

Environment

Disables rendering the *Environment* render pass in the final render.

Surfaces

Disables rendering object materials in the final render.

Curves

Disables rendering curve strands in the final render.

Volume

Disables rendering [Volumes](#) in the final render.

Use

Motion Blur

Render motion blur for this Layer, if enabled in the [Render Settings](#).

Override

Material Override

Overrides all materials in the render layer.

World Override

Overrides world background in the render layer.

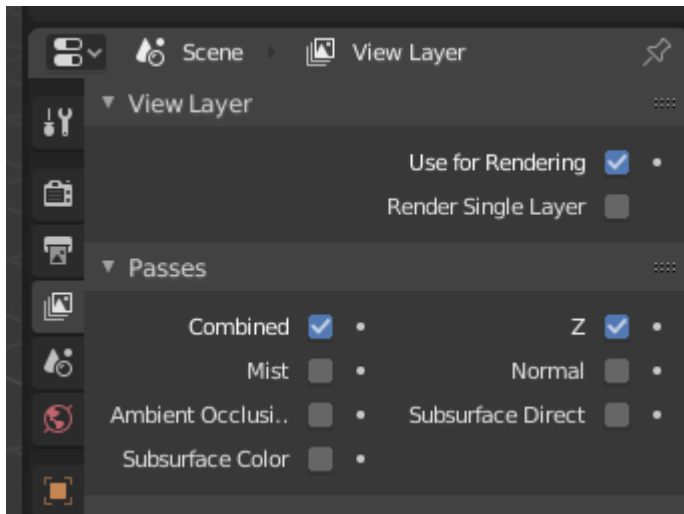
Samples

View layer samples to override the scene samples. Controlled by the [layer samples](#) in the Sampling panel.

View Layer

Reference

Panel:: *Properties ▶ Scene ▶ View Layer*



View Layer panel (shown here for the EEVEE render engine).

The Layer Panel shows the settings of the active View Layer.

Use for Rendering

The active view layer will be used during rendering.

Render Single Layer

Only render the active view layer.

Note

This option is ignored when rendering from the command line.

See also

Additional options shown in this panel are different for each render engine. See [Render Passes](#) for the options per render engine.

Passes

Reference

Panel:: *Scene ▶ View Layers ▶ Passes*

Passes can be used to split rendered images into colors, direct and indirect light to edit them individually, and also to extract data such as depth or normals.

Data

Cycles

Include

Combined

The final combination of render passes with everything included.

Z

Distance to any visible surfaces.

Note

The Z pass only uses one sample. When depth values need to be blended in case of motion blur or [Depth of Field](#), use the mist pass.

Mist

Distance to visible surfaces, mapped to the 0.0 - 1.0 range. When enabled, settings are in [World tab](#). This pass can be used in compositing to fade out objects that are farther away.

Position

World position of objects in the scene.

Normal

Surface normal used for shading.

Vector

Motion vectors for the Vector Blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

UV

Mapped UV coordinates, used to represent where on a mesh a texture gets mapped too. This is represented through the red and green channels of the image. The blue channel is encoded with a constant value of 1 but does not hold any information.

Denoising Data

Includes *Denoising Albedo*, *Denoising Normal*, and a render pass of the original combined pass before denoising.

Note

The Z, Position, Object Index, and Material Index passes are not anti-aliased.

Indexes

Object Index

Creates a mask of the object that can be later read by the [ID Mask Node](#) in the Compositor.

Material Index

Creates a mask of the material that can be later read by the [ID Mask Node](#) in the Compositor.

Debug

Sample Count

Number of samples per pixel taken, divided by the maximum number of samples. To analyze adaptive sampling.

Alpha Threshold

Z, Index, normal, UV and vector passes are only affected by surfaces with alpha transparency equal to or higher than this threshold. With value 0.0 the first surface hit will always write to these passes, regardless of transparency. With higher values surfaces that are mostly transparent can be skipped until an opaque surface is encountered.

EEVEE

Include

Combined

The final combination of render passes with everything included.

Z

Distance to any visible surfaces.

Mist

Distance to visible surfaces, mapped to the 0.0 - 1.0 range.

Normal

Surface normal used for shading.

Position

World position of objects in the scene.

Vector

Motion vectors for the Vector Blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

Light

Cycles

Diffuse

Direct

Direct lighting from diffuse and subsurface BSDFs. We define direct lighting as coming from lights, emitting surfaces, the background, or ambient occlusion after a single reflection or transmission off a surface. BSDF color is not included in this pass.

Indirect

Indirect lighting from diffuse and subsurface BSDFs. We define indirect lighting as coming from lights, emitting surfaces or the background after more than one reflection or transmission off a surface. BSDF color is not included in this pass.

Color

Color weights of diffuse and subsurface BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Glossy

Direct, Indirect, Color

Same as above, but for glossy BSDFs.

Transmission

Direct, Indirect, Color

Same as above, but for transmission BSDFs.

Volume

Direct, Indirect

Same as above, but for volumetric BSDFs.

Other

Emission

Emission from directly visible surfaces.

Environment

Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

Ambient Occlusion

Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a 'normalized' value between 0 and 1.

Shadow Catcher

Extra indirect light information collected by objects with the [Shadow Catcher](#) option enabled. Multiply this pass with existing footage using the [Mix Node](#) in the [Compositor](#) to add the indirect lighting information to the footage.

Note

[Transparent BSDFs are given special treatment](#). A fully transparent surface is treated as if there is no surface there at all; a partially transparent surface is treated as if only part of the light rays can pass through. This means it is not included in the Transmission passes; for that a glass BSDF with index of refraction 1.0 can be used.

EEVEE

Diffuse

Light

Direct lighting from diffuse BSDFs. We define lighting as coming from lights, the background, or ambient occlusion off a surface. BSDF color is not included in this pass.

Color

Color weights of diffuse BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Specular

Light, Color

Same as above, but for specular BSDFs.

Volume

Light

The scattering pass from volume objects or world.

Other

Emission

Emission from directly visible surfaces.

Environment

Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

Shadow

Shadows from light objects. Mostly useful for compositing objects with shadow into existing footage.

Ambient Occlusion

Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a 'normalized' value between 0 and 1.

Transparent

Contain [Blended](#) surfaces, so they can be adjusted in the compositor and later mixed with opaque passes.

This pass only supports monochromatic opacity. Colored opacity will show differently than in combined pass.

Occlusions Distance

Distance of object that contributes to the ambient occlusion effect.

Cryptomatte

Cryptomatte is a standard to efficiently create mattes for compositing. Cycles outputs the required render passes, which can then be used in the

Blender Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing. The mattes will be anti-aliased and take into account effects like motion blur and transparency.

Object

Render cryptomatte object pass, for isolating objects in compositing.

Material

Render cryptomatte material pass, for isolating materials in compositing.

Asset

Render cryptomatte asset pass, for isolating groups of objects with the same [parent](#) in compositing.

Levels

Sets how many unique objects can be distinguished per pixel.

Typical Workflow

1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and link the Render Layer matching Image and Cryptomatte passes to it.
3. Attach a Viewer node to the Pick output of the Cryptomatte node.
4. Use the Cryptomatte Add/Remove button to sample objects in the Pick Viewer node.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

See also

[Cryptomatte Node](#).

Shader AOV

Shader AOVs (Arbitrary Output Variables) provide custom render passes for any shader node components. As an artist this can be a good way to fix or tweak fine details of a scene in post-processing. To use Shader AOVs create the pass in the *Shader AOV* panel then reference this pass with the [AOV Output](#) shading node. Shader AOVs can be added or removed in the *Shader AOV* panel. In this panel is a list of all AOV passes; each AOV in the list consists of a *Name* and *Data Type*.

Active AOV Index

The name of the render pass; this is the *Name* that is referenced in the *AOV Output* node. Any names can be used for these passes, as long as they do not conflict with built-in passes that are enabled.

Data Type

Shader AOVs can either express a *Color* or a *Value* output. The *Color* type as the name suggest can be used for a color but also for normals. A *Value* type can be used for any single numerical value.

Light Groups

Cycles only

Light Groups provide render passes that only contains information from the lights within that group. Light Groups can be used to easily tweak the lighting color and intensity of specific lights without having to re-render the scene.

Active Light Group Index

The name of the render pass. This is the name that is used when adding [World Shaders](#), [Lights and Objects](#) to *Light Groups*.

Lightgroup Sync

These operators are available in the menu to the right of the Light Group list.

Add Used Lightgroups

Adds all Light Groups to the View Layer that have been created elsewhere and have lights assigned to them.

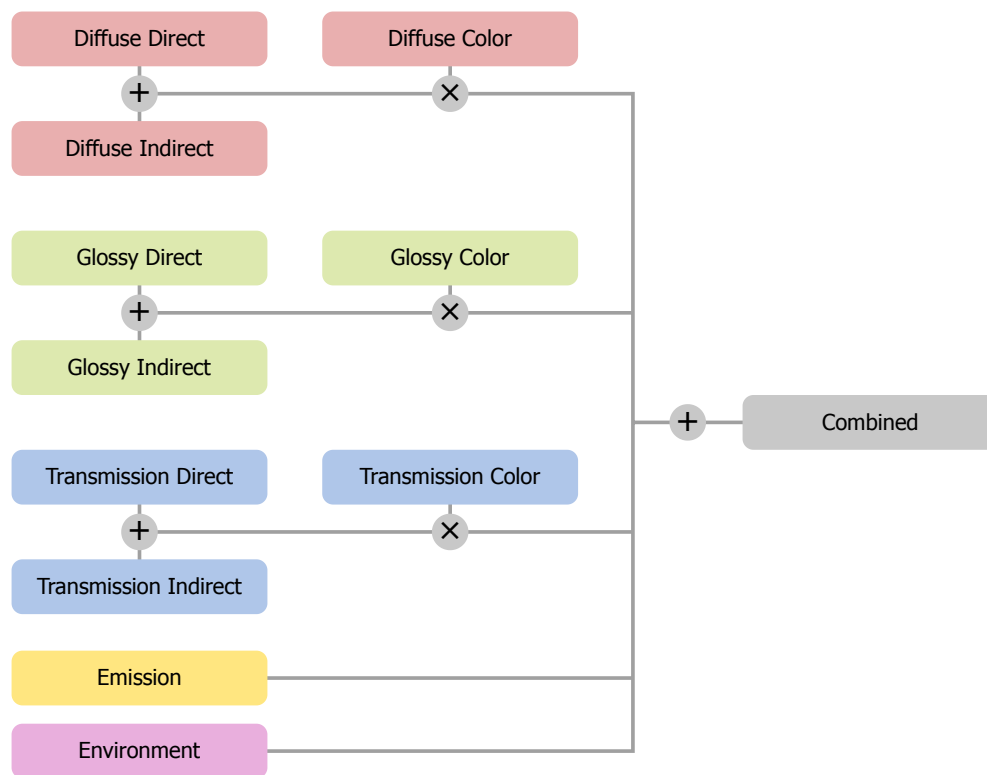
Remove Unused Lightgroups

Deletes all Light Groups that do not have any lights assigned to them.

Combining

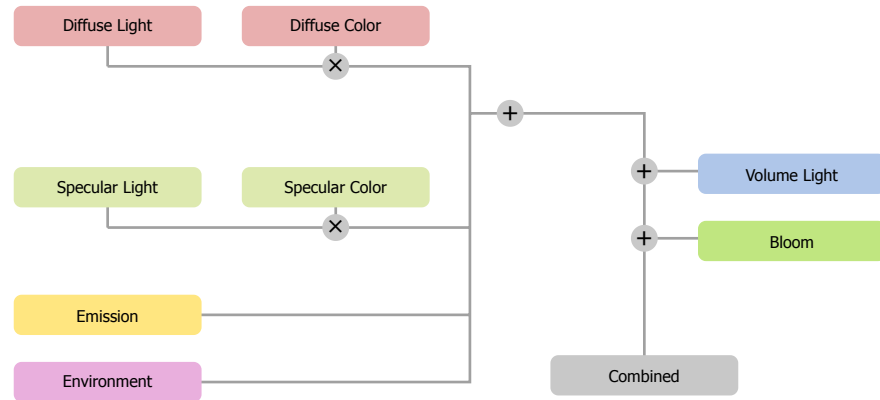
Cycles

All these lighting passes can be combined to produce the final image as follows:



EEVEE

The passes can be combined to produce the final image as follows:



Known Limitations

- Blended materials are not rendered in render passes except the combined pass. Use the *Dithered* as [Render Method](#) to render transparent materials in render passes.
- Depth of field is not rendered in render passes except the combined pass. It is possible to add the depth of field back in the Compositor using the [Defocus node](#).
- EEVEE render passes exclude parts of the BSDF equation. [Shader to RGB](#) is not supported as it needs the full BSDF equation.
- EEVEE supports upto 16 color and 16 value AOV render passes.

Render Output

- [Introduction](#)
- [Output Properties](#)
 - [Format](#)
 - [Frame Range](#)
 - [Stereoscopy](#)
 - [Output](#)
 - [Metadata](#)
 - [Post Processing](#)
- [Audio Rendering](#)
 - [Introduction](#)
 - [Speaker](#)
- [Rendering Animations](#)
 - [Workflow](#)
 - [Frame Sequence Workflow](#)
 - [Hints](#)
- [Animation Player](#)
 - [Player Options](#)
 - [Hotkeys](#)
 - [Frame Cache](#)

Introduction

The first step in the rendering process is to determine and set the output settings. This includes render size, frame rate, pixel aspect ratio, output location, and file type.

When rendering a single frame, the output should be a single image format and not a video. Several [image formats](#) are available.

Images can also be used for rendering animations which has a couple advantages. For example, when rendering animations to image file formats the render job can be canceled and resumed at the last rendered frame by changing the frame range. This is useful if the animation takes a long time to render and the computers resources are needed for something else.

Images can then be encoded to a video by adding the rendered image sequence into the [Video Sequencer](#) and choosing an appropriate [Video Output](#).

Tip

Rendered image sequences can be played back in the [Animation Player](#).

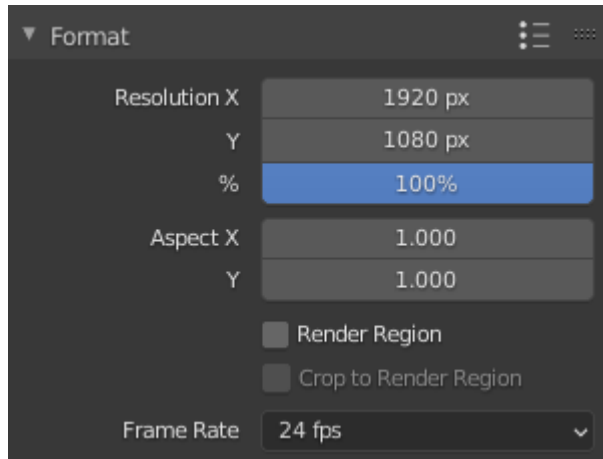
See also

See [Render Menu](#)

Output Properties

- [Format](#)
- [Frame Range](#)
 - [Time Stretching](#)
- [Stereoscopy](#)
 - [Introduction](#)
 - [Usage](#)
- [Output](#)
 - [Color Management](#)
 - [Encoding](#)
- [Metadata](#)
 - [Note](#)
 - [Burn into Image](#)
- [Post Processing](#)

Format



Format panel.

Several render presets exist with common resolution and frame rates for TVs and screens can be selected in the panel header.

Resolution X, Y

The number of pixels horizontally and vertically in the image.

Percentage

Slider to reduce or increase the size of the rendered image relative to the *Resolution* values. This is useful for small test renders that have the same proportions as the final image.

Aspect X, Y

Older televisions may have non-square pixels, so this can be used to control the shape of the pixels along the respective axis. This will *pre-distort* the images which will look stretched on a computer screen, but which will display correctly on a TV set. It is important that you use the correct pixel aspect ratio when rendering to prevent re-scaling, resulting in lowered image quality.

Render Region

Renders just a portion of the view instead of the entire frame. See the [Render Region](#) documentation to see how to define the size of the render region.

Crop to Render Region

Crops the rendered image to the size of the render region, instead of rendering a transparent background around it.

Frame Rate

The number of frames that are displayed per second, relevant for [Animation](#). The menu gives several common frame rates, custom frame rates can be used by selecting *Custom* which gives access to the following properties:

FPS

The frame rate, expressed in frames per second.

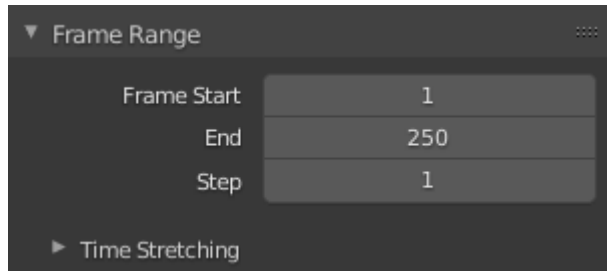
Base

Some standards require a more precise frame rate, for example NTSC. These can be represent as a fraction where the *Base* value is used as the fraction's denominator and the FPS being the numerator: $\frac{\text{FPS}}{\text{Base}}$.

See also

[Viewport Playback Frame Rate Limited](#)

Frame Range



Frame Range panel.

This panel defines how long an animation will last in terms of frames. Frames can be divided by the scene's [Frame Rate](#) to get the animation duration in terms of time. For example, a 250 frame animation at a frame rate of 30 will last 8.3 seconds.

Frame Start, End

Set the *Start* and *End* frames for [Rendering Animations](#).

Step

Controls the number of frames to advance by for each frame in the timeline.

Time Stretching

Use to remap the length of an animation; making it run slower or faster. The *Old* and *New* settings may either be used as absolute values or as a ratio: For example, setting *Old* to a value of 2 and *New* to 1 will run the animation twice as fast.

Warning

Using *Time Stretching* will not influence the *Start* or *End* frames set above, so make sure that your animation is not cut off or has extraneous still frames at the end.

Old

The length in frames of the original animation.

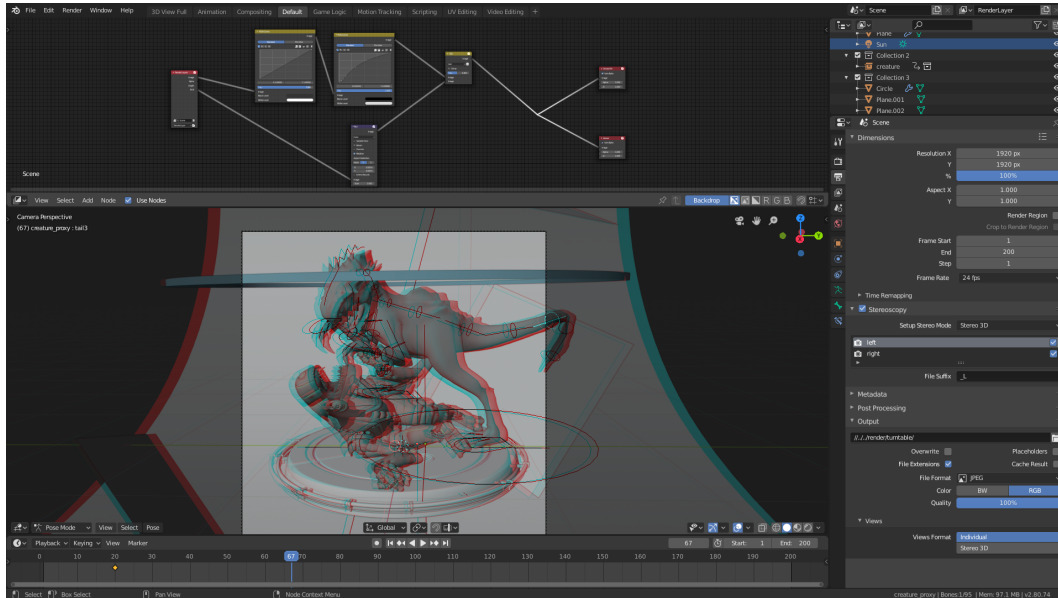
New

The length in frames the new animation will last.

Stereoscopy

- [Introduction](#)
- [Usage](#)
 - [Introduction](#)
 - [Stereoscopy Setup](#)
 - [Camera](#)
 - [Viewport](#)
 - [Stereo 3D Display](#)
 - [Viewport Preview](#)
 - [Rendering and Image Editor](#)
 - [Image Formats](#)
 - [Final Considerations](#)
 - [Window Stereo 3D Display](#)
 - [Stereo 3D Camera](#)
 - [Viewport Stereo 3D](#)
 - [Multi-View and Stereo 3D Image I/O](#)
 - [Image Editor](#)
 - [Compositor](#)

Introduction



Multi-view is a complete toolset for working with stereoscopic rendering in Blender. It works with both the EEVEE and Cycles rendering engines. Cycles additionally supports stereoscopic panoramic cameras. There is support for many different stereo 3D visualization types.

Note

If you have a real 3D display at some point you can change the 3D display mode in the Window menu, by calling the *Stereo 3D* operator. Be aware that some modes require a fullscreen editor to work, and this can be taxing on your CPU.

Usage

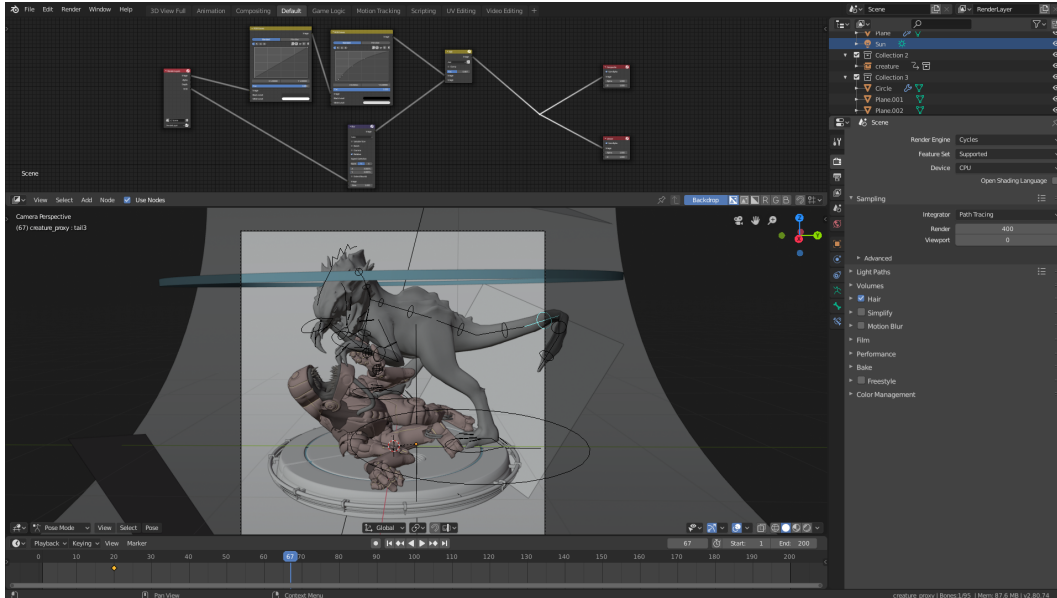
For example, we will take an existing blend-file that was made for monoscopic rendering and transform it to be stereo 3D ready.



Creature Factory 2 by Andy Goralczyk rendered in stereo 3D (anaglyph).

Introduction

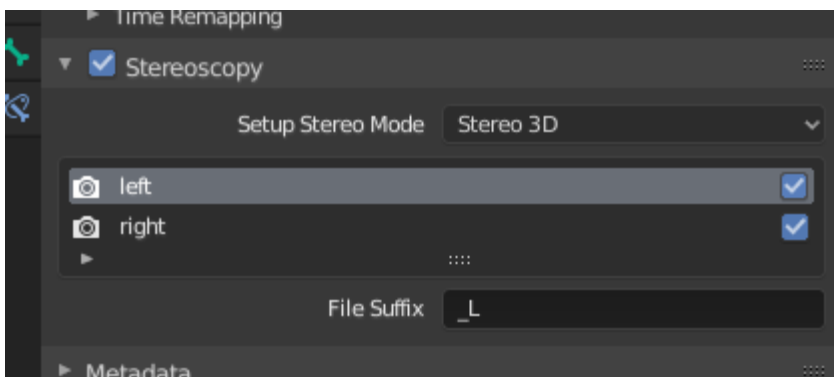
Start opening up your project file, in this case `turntable.blend` from the *Creature Factory 2* Open Movie Workshop series from the Blender Institute by Andy Goralczyk.



Turntable Creature Factory 2.

Stereoscopy Setup

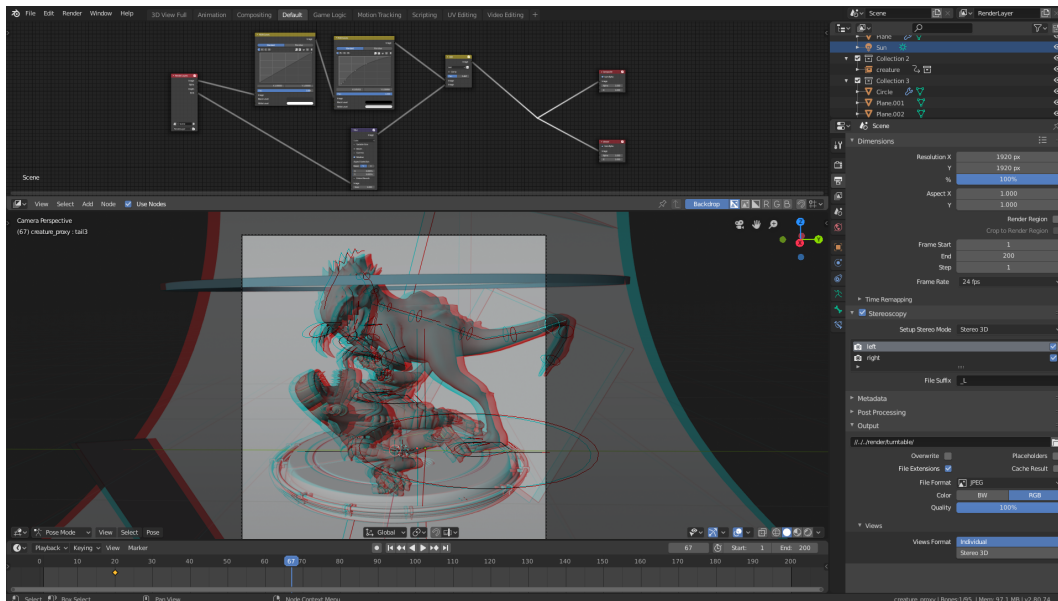
Go to the [Output Properties](#) and enable *Stereoscopy* for this scene.



Scene render views.

Note

When you turn on *Stereoscopy* in the scene, you get 3D preview in the viewport, as well as multiple panels that are now accessible all over the user interface.

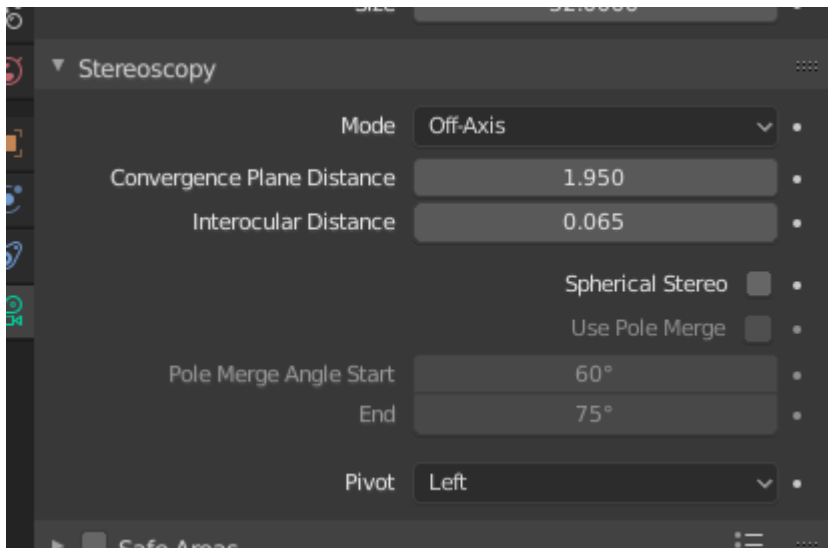


Viewport with 3D visualization.

Camera

To tweak the stereo 3D parameters, select the camera in the Outliner. In the Camera panel go to the Stereoscopy tab and change the *Convergence Distance*.

The viewport will respond in real-time to those changes allowing you to preview the current depth value of the scene.

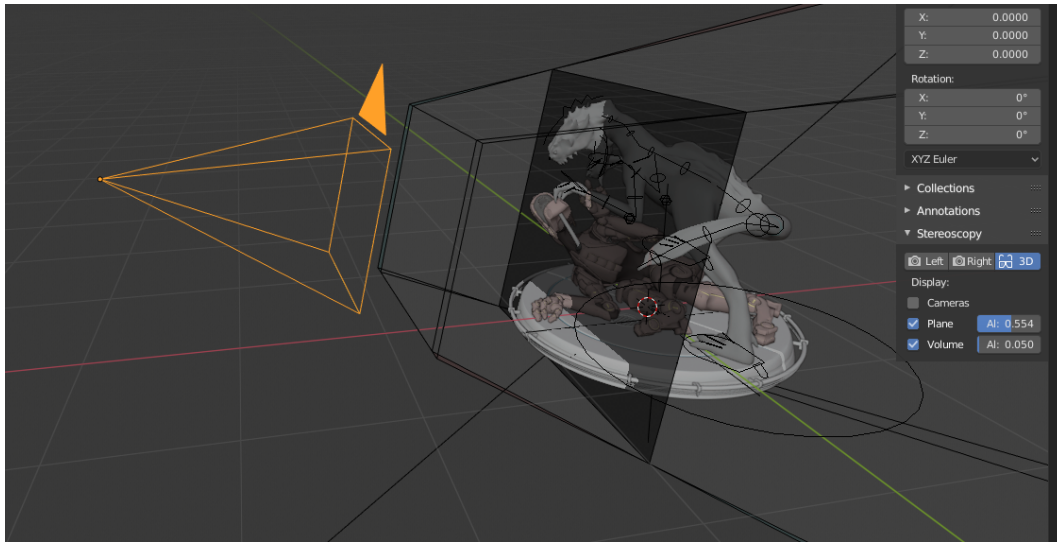


Stereo convergence distance.

Viewport

Before fine-tuning the camera parameters, you can set the convergence plane in the viewport based in your scene depth layout. Go outside the camera view and you will instantly see the convergence plane in front of the camera.

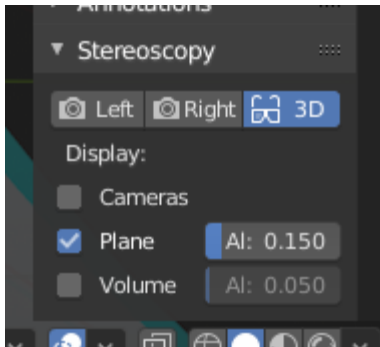
You can toggle this and other display settings in the Stereoscopy panel of the 3D Viewport's Sidebar. In the following image, the camera's frustum volumes are also visible.



Viewport plane and volume stereo preview.

Stereo 3D Display

If you have a real 3D display at some point, you can change the 3D display mode in the Window menu, by calling the Stereo 3D operator. Be aware that some modes require a fullscreen editor to work.

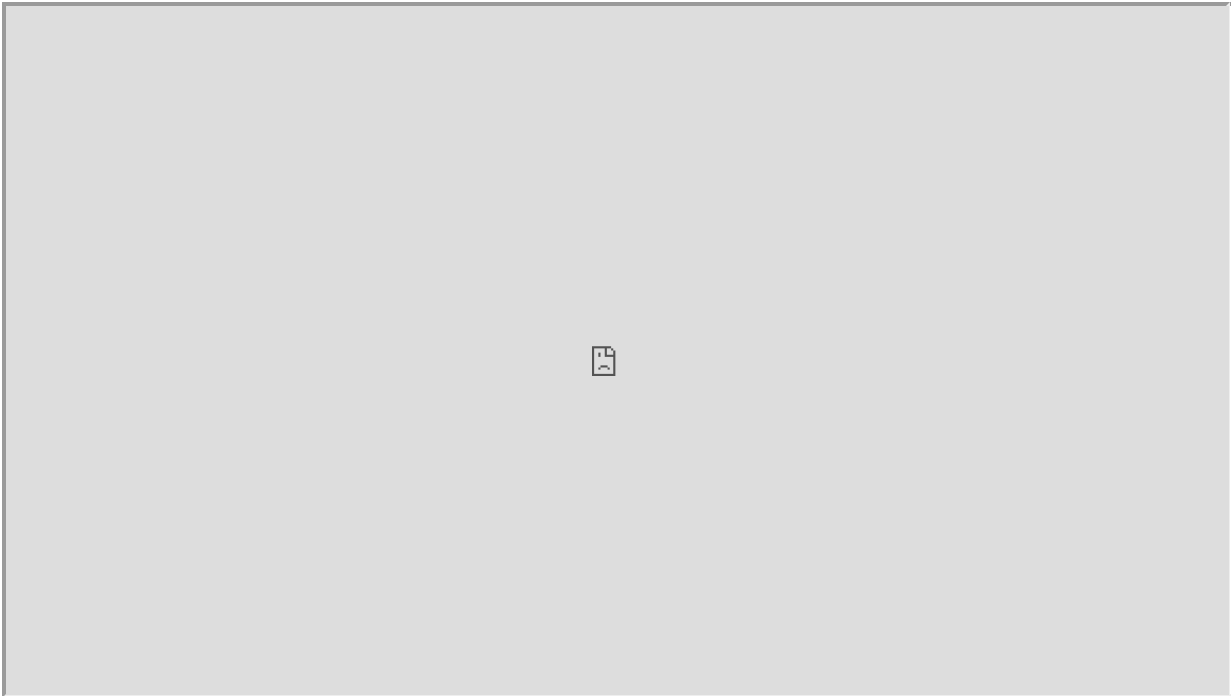


Window menu, stereo 3D operator.

Viewport Preview

Before rendering your scene, you can save a Viewport Preview of the animation for testing in the final display. In the Render Output panel you can choose the output *Views Format*.

The options include individual files per view, top-bottom, anaglyph among others. Pick the one that fits your display requirements.



Rendering and Image Editor

Once you are happy with the results, you can render out the final animation. In the Image Editor you can inspect the individual views and the stereo result.

Image Formats

Your final animation can be saved in more robust formats. In this example we saved as cross-eyed side-by-side stereo 3D.



Side-by-side cross-eye format.

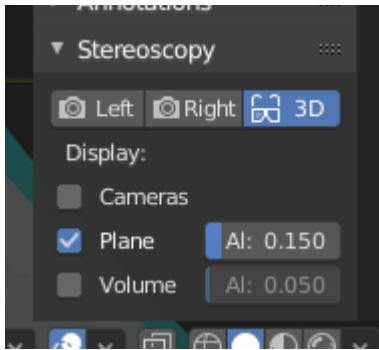
Final Considerations

As this guide showed, there is more to stereo 3D rendering than just generate two images. The earlier the stereo pipeline is considered the smoother it will get. The following sections are a more in-depth view of the individual components we visited in the workflow.

Window Stereo 3D Display

An essential component of the Stereoscopy pipeline is the ability to display the stereo image in a proper display. Blender supports from high-end 3D displays to simple red-cyan glasses. On top of that, you can set a different display mode for each window.

The display mode can be changed via the Window menu or if you create your own shortcuts for the `wm.set_stereo_3d` operator.



Window menu, stereo 3D operator.

Display Mode

Anaglyph

Render two differently filtered colored images for each eye. Anaglyph glasses are required. We support red-cyan, green-magenta and yellow-blue glasses.

Interlace

Render two images for each eye into one interlaced image. A 3D-ready monitor is required. We support Row, Column and Checkerboard Interleaved. An option to Swap Left/Right helps to adjust the image for the screen. This method works better in fullscreen.

Time Sequential

Render alternate eyes. This method is also known as Page Flip. This requires the graphic card to support Quad Buffer and it only works in fullscreen.

Side-by-Side

Render images for left and right eye side-by-side. There is an option to support Cross-Eye glasses. It works only in fullscreen, and it should be used with the Full Editor operator.

Top-Bottom

Render images for left and right eye one above another. It works only in fullscreen, and it should be used with the Full Editor operator.

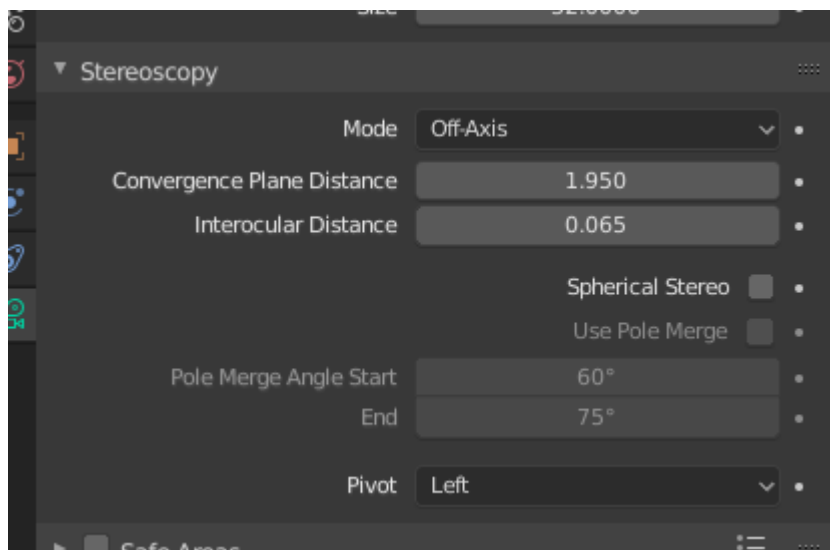
Note

Full Screen Stereo 3D Modes

If you have a 3D display most of the time, you will use it to see in stereo 3D, you will have to go to the fullscreen mode. In fact some modes will only work in the full window mode that hides most of the user interface from the work area. In this case it is recommended to work with two monitors, using the 3D screen for visualizing the stereo result while the other screen can be used for the regular Blender work.

Stereo 3D Camera

When using the Stereo 3D scene view setup, a stereo pair is created on-the-fly and used for rendering and previsualization. For all the purposes this works as two cameras that share most parameters (focal length, clipping, ...). The stereo pair, however, is offset, and can have unique rotation and shift between itself.



Stereo 3D camera settings.

Interocular Distance

Set the distance between the camera pair. Although the convergence of a stereo pair can be changed in post-production, different interocular distances will produce different results due to the parts of the scene being occluded from each point of view.

Convergence Plane Distance

The converge point for the stereo cameras. This is often the distance between a projector and the projection screen. You can visualize this in the 3D Viewport.

Spherical Stereo

Render every pixel rotating the camera around the middle of the interocular distance.

Use Pole Merge

Fade interocular distance to 0 after the given cutoff angle.

Pole Merge Start Angle

Angle at which interocular distance starts to fade to 0.

Pole Merge End Angle

Angle at which interocular distance is 0.

Convergence Mode

Off-Axis

The stereo camera pair is separated by the interocular distance, and shifted inwards so it converges in the convergence plane. This is the ideal format since it is the one closest to how the human vision works.

Parallel

This method produces two parallel cameras that do not converge. Since this method needs to be manually converged it cannot be used for viewing. This method is common when combining real footage with rendered elements.

Toe-in

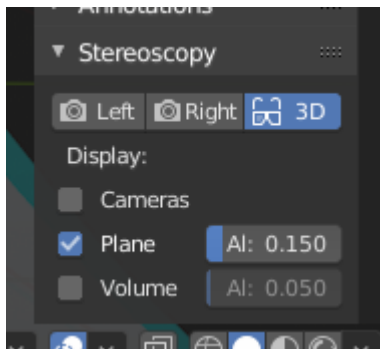
A less common approach is to rotate the cameras instead of shifting their frustum. The Toe-in method is rarely used in modern 3D productions.

Pivot

The stereo pair can be constructed around the active camera with a new camera built for each eye (Center Pivot) or using the existing camera and creating (Left or Right). The latter is what is used when only one eye needs to be rendered for an existing mono 2D project.

Viewport Stereo 3D

When you enable *Views* in the Render Layer panel, a new area is available in the 3D Viewport Sidebar region. In this panel you can pick whether to see the stereo 3D in the viewport, or which camera to see. It also allow you to see the *Cameras*, the *Plane* and the *Volume* of the stereo cameras.



Viewport stereo 3D settings.

Cameras

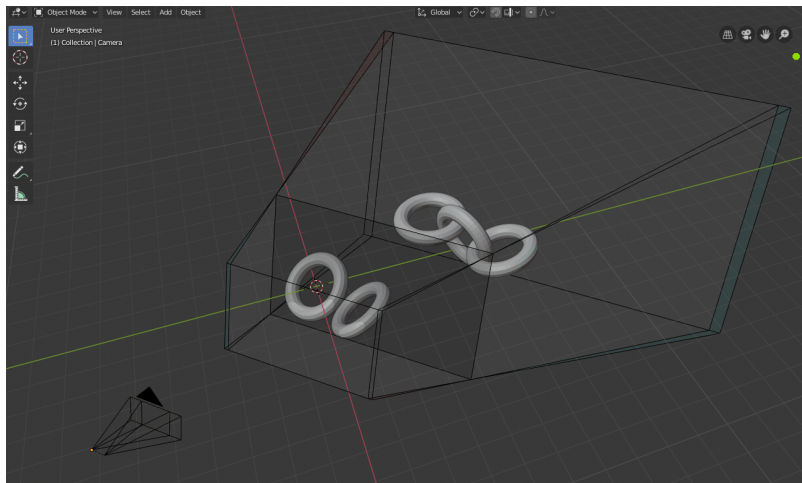
When working with the Stereo 3D Viewports setup, you can inspect what each individual generated camera is looking or the combined result of them. In the Multi-View mode you can see the combined result of the left and right cameras (when available) or the current selected camera.

Plane

The convergence plane represents the screen as it is perceived by the audience. Visualizing it in the 3D Viewport allows you to layout your scene based on your depth script outside the camera view.

Volume

The intersection of the stereo cameras frustums helps planning the show by avoiding elements being visible by only one camera. The volume is defined by the camera's start and end clipping distances. The areas that are in the frustum of one camera only are known as *retinal rivalry areas*. They are tolerated in the negative space (the region from the convergence plane into the image) but are to be avoided at all costs in the positive space (the area from the convergence plane to the camera).



Viewport 3D: convergence plane and volume display.

Multi-View and Stereo 3D Image I/O

Multi-View and Stereo 3D

Multi-view images can be saved in special formats according to the production requirements. By default the system saves each view as an individual file, thus generating as many files as views to be rendered. In stereo 3D productions, for the final deployment or even intermediary previews it is convenient to save stereo 3D images, that are ready to use with 3D displays or simple anaglyph glasses. The formats supported match the display modes available for the window.

Lossy-Formats

Some stereo 3D formats represent a considerable loss of data. For example, the Anaglyph format will cap out entire color channels from the original image. The Top-Bottom compressed will discard half of your vertical resolution data. The Interlace will mash your data considerably. Once you export in those formats, you can still import the image back in Blender, for it to be treated as Stereo 3D. You will need to match the window stereo 3D display mode to the image stereo 3D format though.

Lossless Formats

Some formats will preserve the original data, leading to no problems on exporting and importing the files back in Blender. The Individual option will produce separate images that (if saved in a lossless encoding such as PNG or OpenEXR) can be loaded back in production with no loss of data. For the Stereo 3D formats the only lossless options are *Top-Bottom* and *Side-by-Side* without the Squeezed Frame option.

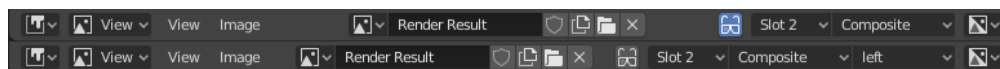
Multi-View OpenEXR

Another option is to use multi-view OpenEXR files. This format can save multiple views in a single file and is backward compatible with old OpenEXR viewers (you see only one view though). Multi-view native support is only available to OpenEXR.

Image Editor

View Menu

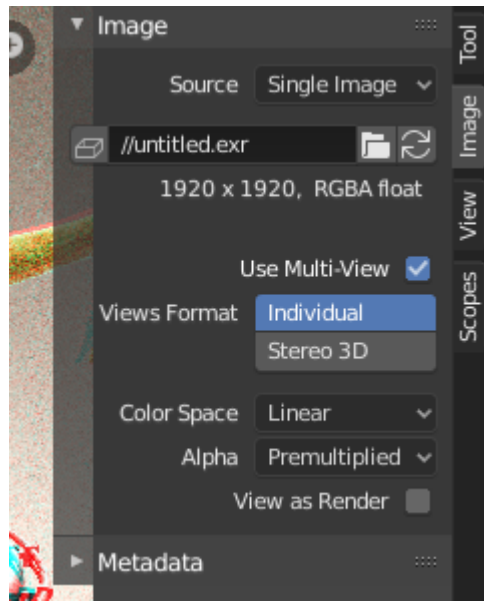
After you render your scene with Stereo 3D you will be able to see the rendered result in the combined stereo 3D or to inspect the individual views. This works for Viewer nodes, render results or opened images.



Stereo 3D and view menu.

Views Format

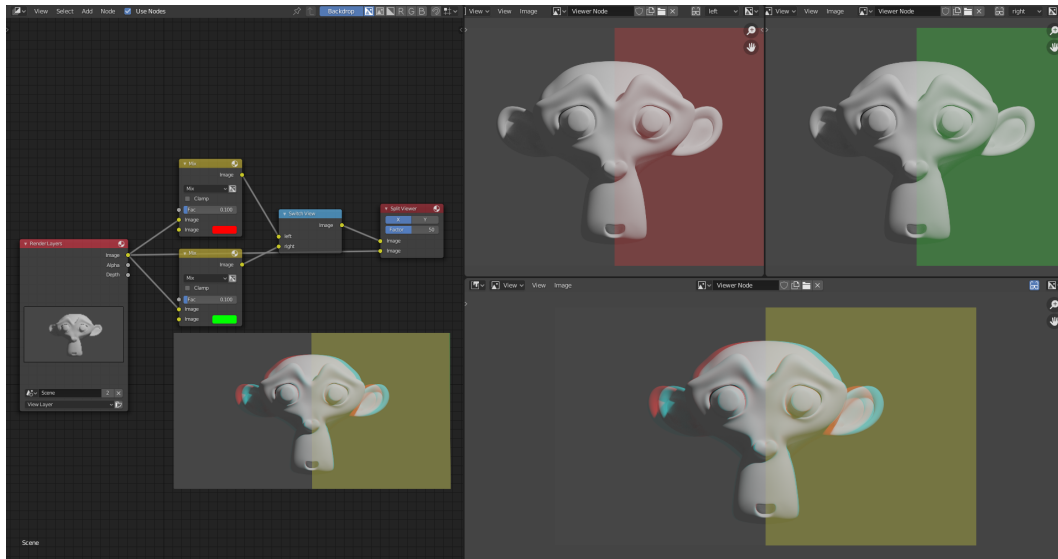
When you drag and drop an image into the Image Editor, Blender will open it as an individual images at first. If your image was saved with one of the Stereo 3D formats, you can change how Blender should interpret the image by switching the mode to Stereo 3D, turning on *Use Multi-View* and picking the corresponding stereo method.



Views formats and stereo 3D.

Compositor

The Compositor works smoothly with multi-view images. The compositing of a view is completed before the remaining views start to be composited. The pipeline is the same as the single-view workflow, with the difference that you can use images, movies or image sequences in any of the supported multi-view formats.



Compositor, backdrop and Split Viewer node.

The views to render are defined in the current scene views, in a similar way as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Note

Single-View Images

If the image from an Image node does not have the view you are trying to render, the image will be treated as a single-view image.

Switch View Node

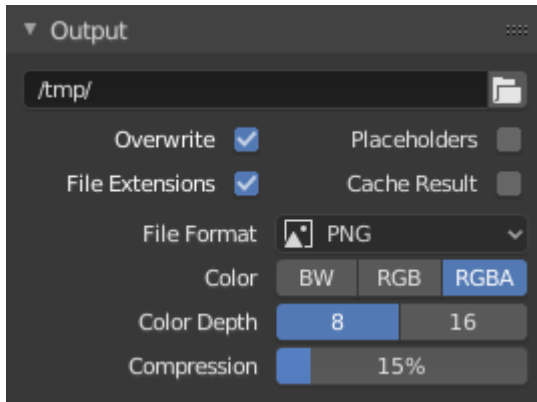
If you need to treat the views separately, you can use the [Switch View node](#) to combine the views before an Output node.

Tip

Performance

By default, when compositing and rendering from the user interface all views are rendered and then composited. During test iterations you can disable all but one view from the Scene Views panel, and re-enable it after you get the final look.

Output



Output panel.

This panel provides options for setting the location of rendered frames for animations, and the quality of the saved images.

File Path

Choose the location to save rendered frames.

When rendering an animation, the frame number is appended at the end of the file name with four padded zeros (e.g. image0001.png). You can set a custom padding size by adding the appropriate number of # anywhere in the file name (e.g. image_##_test.png translates to image_01_test.png).

This setting expands [Relative Paths](#) where a // prefix represents the directory of the current blend-file.

Saving

File Extensions

Adds the correct file extensions per file type to the output files.

Cache Result

Saves the rendered view layers and their [passes](#) to a multi-layer OpenEXR image. The Compositor can then use this file to improve performance, especially for heavy compositing.

The image is stored in the *Render Cache* folder as specified in the [File Paths Preferences](#). You can also load it back into the Image Editor's Render Result, even after closing and reopening Blender; see [Open Cached Render](#).

File Format

Choose the file format to save to. Based on which format is used, other options such as channels, bit depth and compression level are available.

For rendering out to images see: [saving images](#), for rendering to videos see the [Encoding](#) panel.

Color

Choose the color format to save the image to. Note that *RGBA* will not be available for all image formats.

BW, RGB, RGBA

Image Sequence

Overwrite

Overwrite existing files when rendering.

Placeholders

Create empty placeholder frames while rendering.

Hint

Primitive Render Farm

An easy way to get multiple machines to share the rendering workload is to:

- Set up a shared directory over a network file system.
- Disable *Overwrite*, enable *Placeholders* in the *Render Output* panel.

- Start as many machines as you wish rendering to that directory.

Color Management

This panel controls how [Color Management](#) is applied when saving images.

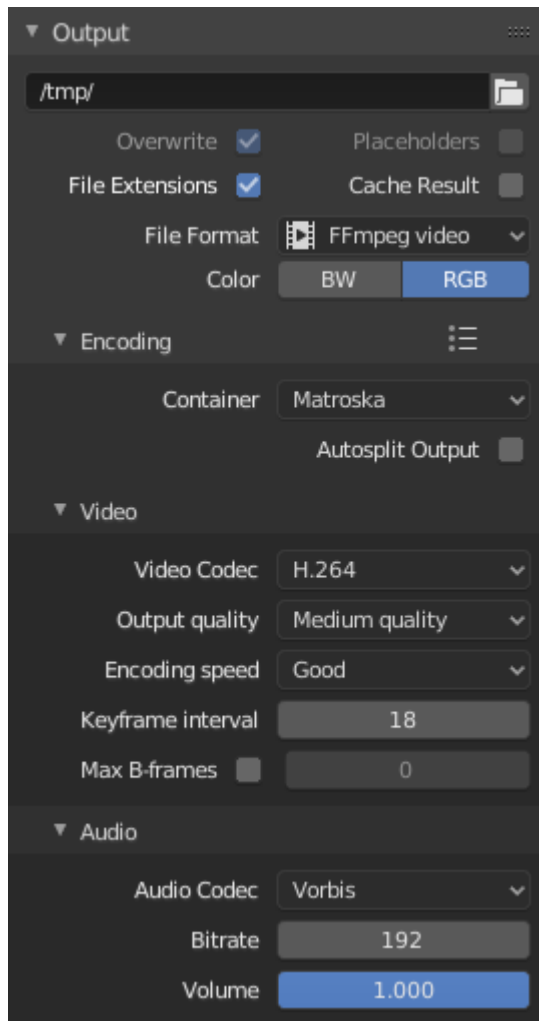
- Follow Scene::** Uses the same color management settings defined by the active Scene. These properties are defined in the Render Settings
- Override::** Uses custom color management settings defined by the properties below in the panel; disregarding any color management settings set at the Scene level.

For a detailed description of color management properties, see the [Color Management](#) page.

Encoding

Reference

Panel:: *Properties ▶ Output ▶ Encoding*



Encoding panel.

Here you choose which video container, codec, and compression settings you want to use. With all of these compression choices, there is a trade-off between file size, compatibility across platforms, and playback quality. In the header, you can use the presets, which choose optimum settings for you for that type of output.

Tip

When you view the [System Console](#), you can see some of the output of the encoding process. You will see even more output if you execute Blender as `blender -d`.

Container

Video container or file type. For a list of all available options, see [video formats](#).

Autosplit Output

If your video is huge and exceeds 2GiB, enable Autosplit Output. This will automatically split the output into multiple files after the first file is 2GiB in size.

Video

Video Codec

Chooses the method of compression and encoding. For a list of all available options see [video formats](#).

Note

Standards

Some containers and codecs are not compatible with each other, so if you are getting errors check that your container and codec are compatible. Like containers and codecs are sometimes not compatible with each other, some codecs do not work with arbitrary dimensions. So, try to stick with common dimensions or research the limitations of the codec you are trying to use.

Output Quality

These are preset [Rate](#).

Encoding Speed

Presets to change between a fast encode (bigger file size) and more compression (smaller file size).

Keyframe Interval

The number of pictures per [Group of Pictures](#). Set to 0 for “intra_only”, which disables [inter-frame](#) video. A higher number generally leads to a

smaller file but needs a higher-powered device to replay it.

Max B-frames

Enables the use of [B-frames](#).

Interval

The maximum number of [B-frames](#) between non-B-frames.

Rate

Bitrate

Sets the average [bit rate](#) (quality), which is the count of binary digits per frame. See also: [FFmpeg -b:v](#).

Minimum / Maximum

Video files can use what is called variable bit rate (VBR). This is used to give some segments of the video less compressing to frames that need more data and less to frames with less data. This can be controlled by the *Minimum* and *Maximum* values.

Buffer

The [decoder bitstream buffer](#) size.

Mux Rate

Maximum bit rate of the multiplexed stream. [Multiplexing](#) is the process of combining separate video and audio streams into a single file, similar to packing a video file and MP3 audio file in a zip-file.

Mux Packet Size

Reduces data fragmentation or muxer overhead depending on the source.

Audio

These settings change how sound is exported while rendering. To control how sound is played back from within Blender, see the audio settings in the [Preferences](#).

Audio Codec

Audio format to use. For a list of all available options, see [video formats](#).

Audio Channels

The number of audio source “locations” to encode.

Mono::	Output a single audio channel.
Stereo::	Output two audio channels; typically a left and right channel.
4 Channels::	Output a four audio channels.
5.1 Surround::	Output a five audio channels with one <u>LFE</u> channel.
7.1 Surround::	Output a seven audio channels with one <u>LFE</u> channel.

Sample Rate

Sets the audio [sampling rate](#).

Bitrate

For each codec, you can control the bit rate (quality) of the sound in the movie. Higher bit rates are bigger files that stream worse but sound better. Use powers of 2 for compatibility.

Volume

Sets the output volume of the audio.

Tips

Tip

The choice of video format depends on what you are planning to do.

It's not recommended to render directly to a video format in the first instance. If a problem occurs while rendering, the file might become unplayable and you will have to re-render all frames from the beginning. If you first render out a set of static images such as the default PNG format or the higher-quality OpenEXR (which can retain HDR pixel data),

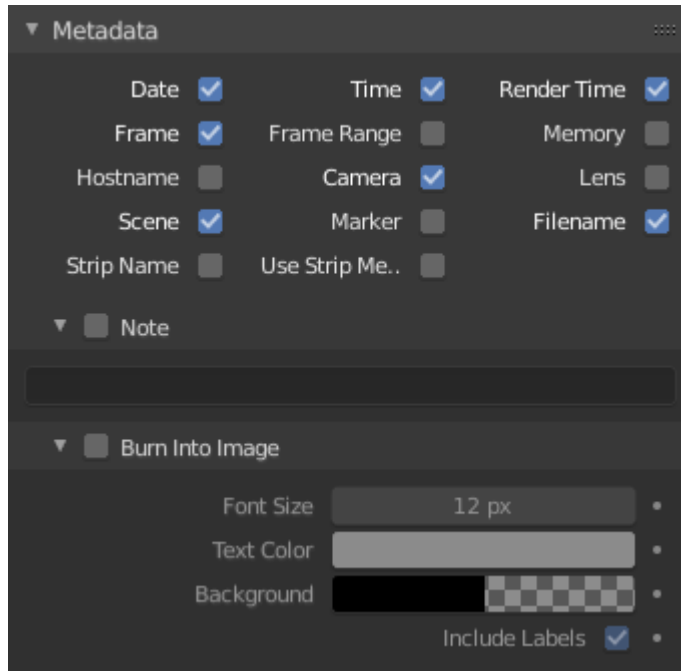
you can combine them as an [Image Strip](#) in the Video Sequencer. This way, you can easily:

- Restart the rendering from the place (the frame) where any problem occurred.
- Try out different video encoding options in seconds, rather than minutes or hours as encoding is usually much faster than rendering the 3D scene.
- Enjoy the rest of the features of the Video Sequencer, such as adding [Image Strips](#) from previous renders, audio, video clips, etc.

Tip

You shouldn't post-process a lossy-compressed file as the compression artifacts may become visible. Lossy compression should be reserved as a final 'delivery format'.

Metadata



Metadata panel.

The *Metadata* panel includes options for writing metadata into render output.

Note

Only some image formats support metadata: See [image formats](#).

Metadata Input

Where to grab metadata from.

Scene:: Use metadata from the current scene.
Sequencer Strips:: Use metadata from the strips in the Sequencer.

Include
Date

Includes the current date and time.

Time

Includes the current scene time and render frame at HH:MM:SS.FF.

Render Time

Includes the render time.

Frame

Includes the frame number.

Frame Range

Includes the start and end frame numbers.

Memory

Includes the peak memory usage.

Hostname

Includes the rendering machine's [hostname](#).

Camera

Includes the name of the active camera.

Lens

Includes the name of the active camera's lens value.

Scene

Includes the name of the active scene.

Marker

Includes the name of the last marker.

Filename

Includes the filename of the blend-file.

Strip Name

Includes the name of the foreground sequence strip.

Note

Includes a custom note.

Hint

It can be useful to use the *Note* field if you are setting up a render farm. Since you can script any information you like into it, such as an identifier for the render node or the job number. For details on stamping arbitrary values, see: [this page](#).

Burn into Image

Add metadata as text to the render.

Font Size

Set the size of the text.

Text Color

Set the color and alpha of the stamp text.

Background

Set the color and alpha of the color behind the text.

Include Labels

Displays the labels before the metadata text. For example, “Camera” in front of the camera name, etc.

Post Processing

Reference

Panel:: *Properties ▶ Output ▶ Post Processing*

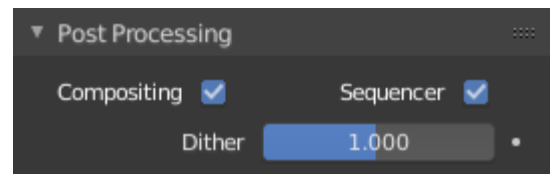
The Post Processing panel is used to control different options used to process your image after rendering.

Pipeline

Todo.

Compositing

Renders the output from the compositing node setup, and then applies the Composite node tree on all images, displaying the image inputted in the Composite Output node.



Post Processing panel.

Sequencer

Renders the output of the Video Sequence editor, instead of the view from the 3D scene's active camera. If the sequence contains Scene strips, these will also be rendered as part of the pipeline. If *Compositing* is also enabled, the Scene strip will be the output of the Compositor.

Dither

Dithering is a technique for blurring pixels to prevent banding that is seen in areas of gradients, where stair-stepping appears between colors. Banding artifacts are more noticeable when gradients are longer, or less steep. Dithering was developed for graphics with low bit depths, meaning they had a limited range of possible colors.

Dithering works by taking pixel values and comparing them with a threshold and neighboring pixels then does calculations to generate the

appropriate color. Dithering creates the perceived effect of a larger color palette by creating a sort of visual color mixing. For example, if you take a grid and distribute red and yellow pixels evenly across it, the image would appear to be orange.

Audio Rendering

- [Introduction](#)
 - [Options](#)
- [Speaker](#)
 - [Options](#)

Introduction

Audio can be rendered from the [Render Menu](#).

Options

Relative Path

Select the file path relative to the blend-file.

Accuracy

Sample accuracy, important for animation data (the lower the value, the more accurate).

Container

See [here](#).

Channels

The number of audio source “locations” to encode. Each channel can be mixed to a separate file by enabling *Split Channels* (see below).

Mono::	Output a single audio channel.
Stereo::	Output two audio channels; typically a left and right channel.
Stereo LFE::	Output a two audio channels with a “low-frequency effects” channel for frequencies below 120.
4 Channels::	Output a four audio channels.
5.1 Surround::	Output a five audio channels with one <u>LFE</u> channel.
6.1 Surround::	Output a six audio channels with one <u>LFE</u> channel.
7.1 Surround::	Output a seven audio channels with one <u>LFE</u> channel.

Format

Some *Audio Containers* also have option to choose a codec. For more information see [here](#).

Sample Rate

Sets the audio [sampling rate](#).

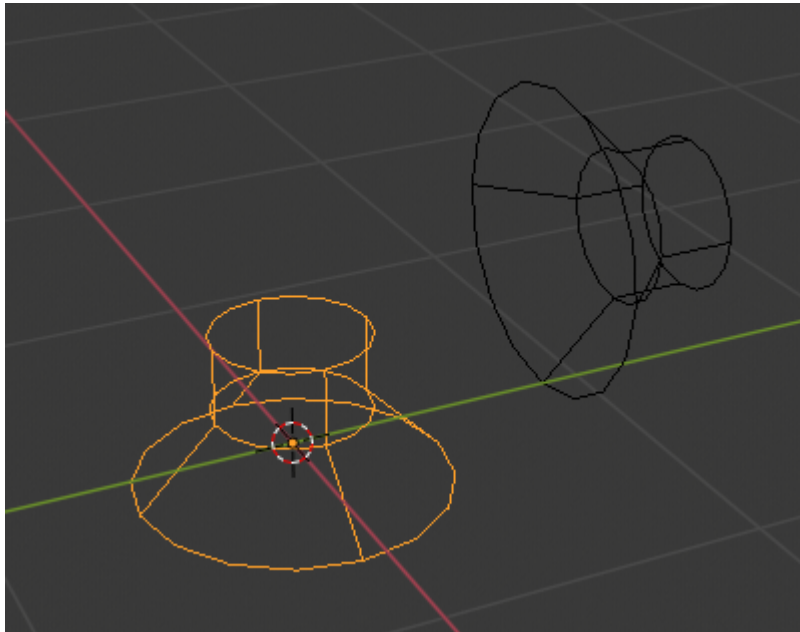
Split Channels

Each audio channel will be rendered into a separate file.

See also

- See [Scene Audio](#) settings.
- See [Audio Output](#) settings.
- See [Audio Preferences](#).

Speaker



Speaker objects.

The speaker object is used to give sound in the 3D Viewport. After adding the object, the various settings can be changed in the Properties.

Options

Sound

Open

The [Data-Block Menu](#) for loading audio files. There are two properties you can check when loading a sound:

Cache

This means the whole sound will be decoded and the raw audio data will be buffered in memory, which results in faster playback, but uses quite a lot of memory. So this should be used for short sound

effects that get played more often, but not for longer audio files such as music.

Mono

For any 3D audio or panning effects the sound source has to be single channel, otherwise it's assumed that the 3D audio and panning information is already present in the multichannel file. Enable this if you want to use those effects for a file with multiple channels.

Mute

Toggles whether or not the sound can be heard.

Volume

Adjust the loudness of the sound.

Pitch

Can be used to bend the pitch of the sound to be either deeper or higher. This basically changes the playback speed of the sound which also results in a pitch change.

Playback Time

There is no setting to choose the start time when the speaker should start playing, because you might want a single speaker to play multiple times. Therefore, you have to open the *NLA Editor* where you can add Sound strips that define when the sound should start (nothing else, so any other properties of the strips, like length don't matter). When you add a speaker object such a strip will be added at the current frame.

Distance

Distance attenuation relevant settings.

Volume

Minimum/Maximum

No matter how far away/close the object is, the distance-based volume won't be lower/higher than this value.

Attenuation

How strong the distance affects the volume. This factor sets the strength of the distance-based volume change, depending on the chosen distance model (see [scene settings](#)).

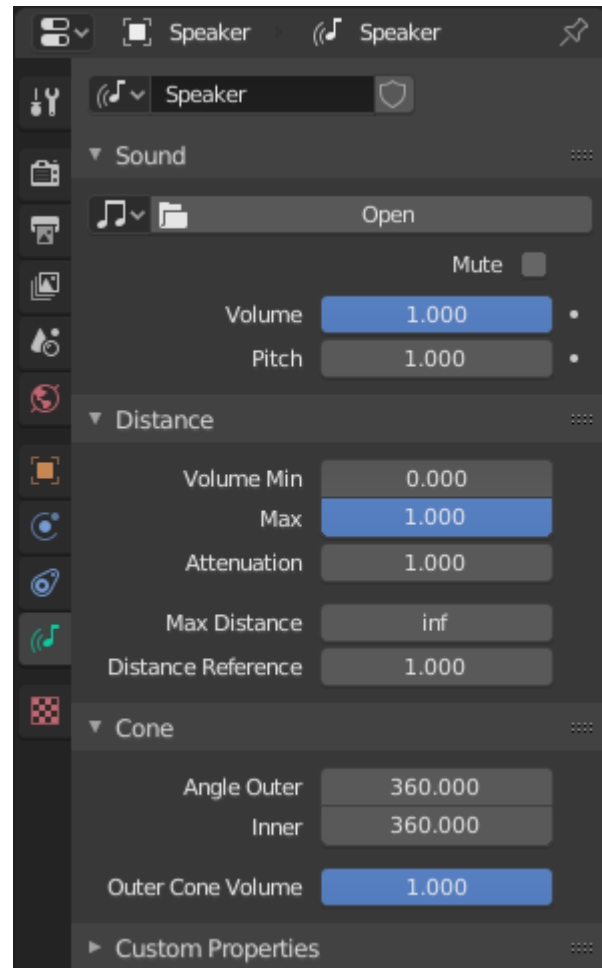
Distance

Maximum

If the object is farther away than this distance, this distance is used to calculate the distance-based volume. Influence of this value also depends on the distance model.

Reference

The distance at which the volume is full (1.0). Set this value to the distance used for recording the sound. Usually sound effects recordings should be made exactly 1 m away from sound to get an accurate volume.



Speaker properties.

Cone

Directionality relevant settings.

Imagine a cone with the top at the original of the speaker object and the main axis of it facing in the same direction as the speaker. There are two cones an inner and an outer cone. The angles represent their opening angles, so 360° mean the cone is fully open and there is no directionality anymore. Inside the inner cone the volume is full (1.0), outside the outer cone the

volume is, whatever one sets for the outer cone volume and the volume between those two cones, linearly interpolated between this two volumes.

Angle

Outer

Angle of the outer cone in degrees. Outside this cone, the volume is equal to the *Outer* volume.

Inner

Angle of the inner cone in degrees. Inside the cone, the volume is full.

Volume

Outer

Volume outside the outer cone.

Rendering Animations

While rendering stills will allow you to view and save the image from the render buffer when it is complete, animations are a series of images, or frames, and are automatically saved directly out to a drive after being rendered.

After rendering the frames, you may need to edit the clips, or first use the Compositor to do green-screen masking, matting, color correction, DOF, and so on to the images. That result is then fed to the Sequencer where the strips are cut and mixed and a final overlay is done.

Finally you can render out from the Sequencer and compress the frames into a playable movie clip.

Workflow

Generally, you do a lot of intermediate renders of different frames in your animation to check for timing, lighting, placement, materials, and so on. At some point, you are ready to make a final render of the complete animation for publication.

There are two approaches you can use when making a movie, or animation, with or without sound. The approach you should use depends on the amount of CPU time you will need to render the movie. You can render a “typical” frame at the desired resolution, and then multiply by the number of frames that will ultimately go into the movie, to arrive at a total render time.

If the total render time is an hour or more, you want to use the “Frame Sequence” approach. For example, if you are rendering a one-minute video clip for film, there will be (60 seconds per minute) X (24 frames per second) or 1440 frames per minute. If each frame takes 30 seconds to render, then you will be able to render two frames per minute, or need 720 minutes (12 hours) of render time.

Rendering takes all available CPU time; you should render overnight, when the computer is not needed, or set Blender to a low priority while rendering, and work on other things (be careful with the RAM space!).

Direct Approach

The Direct Approach, which is highly **not** recommended and not a standard practice, is where you set your output format to an AVI or MOV format, and click *Animation* to render your scene directly out to a movie file. Blender creates one file that holds all the frames of your animation. You can then use Blender's [Video Sequencer](#) to add an audio track to the animation and render out to an MPEG format to complete your movie.

Frame Sequence

The Frame Sequence is a much more stable approach, where you set your output format to a still format (such as JPG, PNG or a multi-layer format). Click *Animation* to render your scene out to a set of images, where each image is a frame in the sequence.

Blender creates a file for each frame of the animation. You can then use Blender's Compositor to perform any frame manipulation (post-processing). You can then use Blender's [Video Sequencer](#) to load that final image sequence, add an audio track to the animation, and render out to an MPEG format to complete your movie. The Frame Sequence approach is a little more complicated and takes more drive space, but gives you more flexibility.

Here are some guidelines to help you choose an approach.

Direct Approach

- Short segments with total render time under one hour.
- Stable power supply.
- Computer not needed for other uses.

Frame Sequence Approach

- Total render time over one hour.
- Post-production work needed:
 - Color/lighting adjustment
 - Green screen/matte replacement
 - Layering/compositing
 - Multiple formats and resolutions of the final product
- Intermediate frames/adjustments needed for compression/codec.
- Precise timing (e.g. lip-sync to audio track) needed in parts.
- May need to interrupt rendering to use the computer, and want to be able to resume rendering where you left off.

Frame Sequence Workflow

1. First prepare your animation.
2. In the *Format* panel, choose the render size, Pixel Aspect Ratio, and the Range of Frames to use, as well as the frame rate, which should already be set.
3. In the Output panel set up your animation to be rendered out as images, generally using a format that does not compromise any quality.
4. Choose the output path and file type in the Output panel as well, for example `//render/my-anim-`.
5. Confirm the range of your animation (frame Start and End).
6. Save your blend-file.
7. Press the *Animation* button and once the animation is finished, use your file manager to navigate to the output folder (render in this example). You will see lots of images that have a sequence number attached to. These are the single frames.
8. In Blender, open the [Video Sequencer](#).

Note

The Video Sequencer does not support multi-layer EXR files. To render to a video format you will have to skip the next three steps and instead use an [Image Input node](#) in the [Compositor](#).

9. Choose *Add Image* from the add menu. Select all the frames from your output folder that you want to include in your animation. They will be added as a strip in the Sequence editor.
10. Now you can edit the strip and add effects or leave it like it is. You can add other strips, like an audio strip.
11. Scrub through the animation to check if you have included all the frames.
12. In the Output panel, choose the container and codec you want (e.g. MPEG H.264) and configure them. The video codecs are described in [Output Options](#).
13. Click the *Animation* render button and Blender will render out the Sequence editor output into a movie.

Hints

Your computer accidentally turns off in the middle of rendering your movie!

Unless your animation renders in a few minutes, it is best to render the animation as separate image files. Instead of rendering directly to a compressed movie file, use a lossless format (e.g. PNG).

This allows you an easy recovery if there is a problem and you have to re-start the rendering, since the frames you have already rendered will still be in the output directory.

Just disable the *Overwrite* option to start rendering where you left off.

You can then make a movie out of the separate frames with Blender's Sequence editor or use 3rd party encoding software.

Animation Preview

It can be useful to render a subset of the animated sequence, since only part of an animation may have an error.

Using an image format for output, you can use the *Frame Step* option to render every *Nth* frame. Then disable *Overwrite* and re-render with *Frame Step* set to 1.

Animation Player

Reference

Menu:: *Topbar ▶ Render ▶ View Animation*

Shortcut:: *Ctrl-F11*

The animation player is a utility typically used for previewing rendered animations, supporting all image and video formats also supported by Blender. This is a convenient way to play back image sequences at the correct frame rate.

Launching the animation player opens a new window, playing back images or a video located at the render output of the current scene. You can also drop images or movie files in a running animation player. It will then restart the player with the new data.

Tip

An external player can also be used instead of the one included in Blender. To do this, select it in the [Preferences](#).

Player Options

Ping Pong

When enabled, playback loops forwards than backwards.

X/Y Flip

Flip the image horizontally or vertically.

Viewing the animation from a different perspective can help you see the animation with “fresh eyes”.

Hotkeys

The following table shows the available hotkeys for the animation player.

Playback

Action	Hotkey
Start/Pause:	Spacebar
Start playback (when paused):	Return
Quit:	Esc

Timeline

Action	Hotkey
Scrub in time:	LMB
Step back one frame:	Left
Step forward one frame:	Right
Step back 10 frames:	Down

Action	Hotkey
Step forward 10 frames:	Up
Manual frame stepping:	NumpadPeriod

Playback Options

Action	Hotkey
Backward playback:	Shift-Down
Forward playback	Shift-Up
Slow down playback:	NumpadMinus
Speed up playback:	NumpadPlus
Toggle looping:	Numpad0
Toggle frame skipping:	A
Toggle ping-pong:	P

Display

Action	Hotkey
--------	--------

Toggle Playhead (Indicator):	I
------------------------------	---

Flip image on the X axis:	F
---------------------------	---

Flip image on the Y axis:	Shift-F
---------------------------	---------

Hold to show frame numbers:	Shift
-----------------------------	-------

Zoom in:	Ctrl-NumpadPlus
----------	-----------------

Zoom out:	Ctrl-NumpadMinus
-----------	------------------

Frame Rate

Action	Hotkey
--------	--------

60 fps	Numpad1
--------	---------

50 fps	Numpad2
--------	---------

30 fps	Numpad3
--------	---------

Action Hotkey

25 fps	Numpad4
--------	---------

24 fps	Shift-Numpad4
--------	---------------

20 fps	Numpad5
--------	---------

15 fps	Numpad6
--------	---------

12 fps	Numpad7
--------	---------

10 fps	Numpad8
--------	---------

6 fps	Numpad9
-------	---------

5 fps	NumpadSlash
-------	-------------

Frame Cache

Image files are cached during playback for faster access.

While loading images is rarely a bottleneck, there are situations where high resolution images may slow down playback causing frame skipping.

See also

[Memory Cache Limit](#) preference to control this limit, which may be increased to cache more images during playback. [Animation Playback Options](#) to specify this value when launching from the command line.

Compositing

- [Introduction](#)
 - [Getting Started](#)
 - [Examples](#)
 - [Image Size](#)
 - [Saving your Composite Image](#)
- [Sidebar](#)
 - [View](#)
 - [Options](#)
- [GPU Compositor](#)
 - [Data](#)
 - [Compositing Space](#)
 - [Output](#)

Node Types

- [Input Nodes](#)
- [Output Nodes](#)

-
- [Color Nodes](#)
 - [Filter Nodes](#)

-
- [Keying Nodes](#)
 - [Mask Nodes](#)

-
- [Tracking Nodes](#)

-
- [Transform Nodes](#)
 - [Utilities Nodes](#)

- [Vector Nodes](#)
-

- [Group](#)
- [Layout Nodes](#)

Introduction

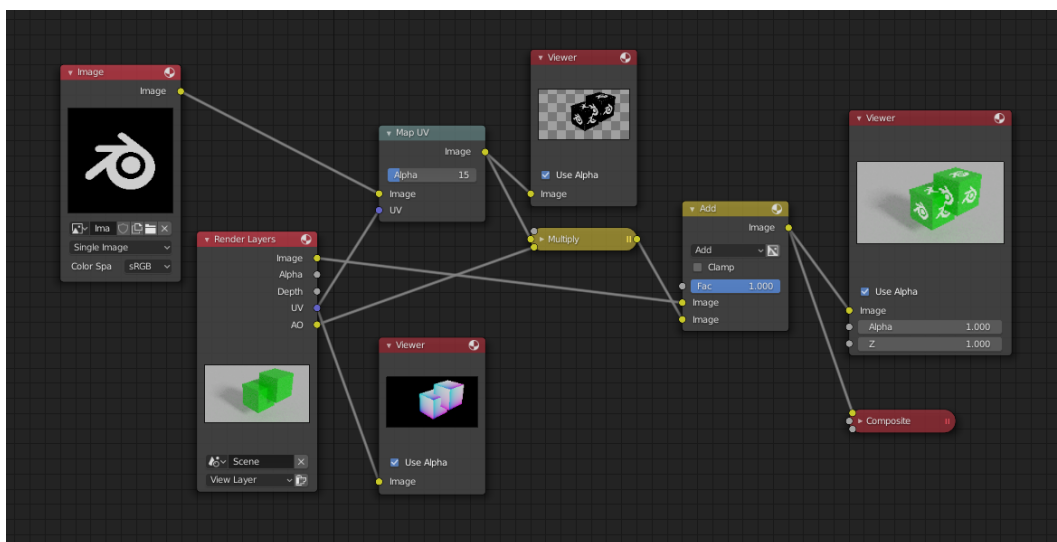
Compositing Nodes allow you to assemble and enhance an image (or movie). Using composition nodes, you can glue two pieces of footage together and colorize the whole sequence all at once. You can enhance the colors of a single image or an entire movie clip in a static manner or in a dynamic way that changes over time (as the clip progresses). In this way, you use composition nodes to both assemble video clips together and enhance them.

Note

Term: Image

The term *Image* may refer to a single picture, a picture in a numbered sequence of images, or a frame of a movie clip. The Compositor processes one image at a time, no matter what kind of input you provide.

To process your image, you use nodes to import the image into Blender, change it, optionally merge it with other images, and finally, save it.



An example of a composition.



An example of color correction.

Getting Started

Access the *Compositor* and activate nodes for compositing by clicking the *Use Nodes* checkbox in the header (see [Introduction](#)).

Note

After clicking *Use Nodes* the Compositor is enabled, however, it can also be disabled in the [Post Processing](#).

You now have your first node setup, from here you can add and connect many types of [Compositing Nodes](#), in a sort of map layout, to your heart's content (or physical memory constraints, whichever comes first).

Note

Nodes and node concepts are explained in more detail in the [Nodes](#) reference.

Examples

You can do just about anything with images using nodes.

Raw footage from a foreground actor in front of a blue screen, or a rendered object doing something, can be layered on top of a background. Composite both together, and you have composited footage.

You can change the mood of an image:

- To make an image ‘feel’ colder, a blue tinge is added.
- To convey a flashback or memory, the image may be softened.
- To convey hatred and frustration, add a red tinge or enhance the red.
- A startling event may be sharpened and contrast-enhanced.
- To convey a happy feeling add yellow (equal parts red and green, no blue).
- Dust and airborne dirt are often added as a cloud texture over the image to give a little more realism.

Image Size

It is recommended to pay attention to image resolution and color depth when mixing and matching images. [Aliasing](#), color *flatness*, or distorted images can all be traced to mixing inappropriate resolutions and color depths.

The Compositor can mix images with any size, and will only perform operations on pixels where images have an overlap. When nodes receive inputs with differently sized Images, these rules apply:

- The first/top Image input socket defines the output size.

- The composite is centered by default, unless a translation has been assigned to a buffer using a *Translate* node.

So each node in a composite can operate on different sized images as defined by its inputs. Only the *Composite* output node has a fixed size, as defined by the [Format Output Properties](#). The *Viewer* node always shows the size from its input, but when not linked (or linked to a value) it shows a small 320×256 pixel image.

Saving your Composite Image

The *Render* button renders a single frame or image. Save your image using [Save Image](#). The image will be saved using the image format settings on the Render panel.

To save a sequence of images, for example, if you input a movie clip or used a Time node with each frame in its own file, use the *Animation* button and its settings. If you might want to later overlay them, be sure to use an image format that supports an Alpha channel (such as PNG). If you might want to later arrange them front to back or create a depth of field effect, use a format that supports a Z-depth channel (such as EXR).

To save a composition as a movie clip (all frames in a single file), use an AVI or QuickTime format, and use the *Animation* button and its settings.

Sidebar

View

Reference

Panel:: *Sidebar region ▶ View*

Backdrop

The backdrop is the output of a Viewer node in the background of the Compositor. For example, when Shift-Ctrl-LMB on an Image node, it displays a preview of the image, or on a Mix node, will show the result of the mixing. You can toggle the backdrop by clicking the checkbox in the *Backdrop* panel title or by clicking on the *Backdrop* button in the header.

Channels

The color channels to display of the backdrop image.

Zoom Alt-V V

Sets the size of the backdrop image.

Offset

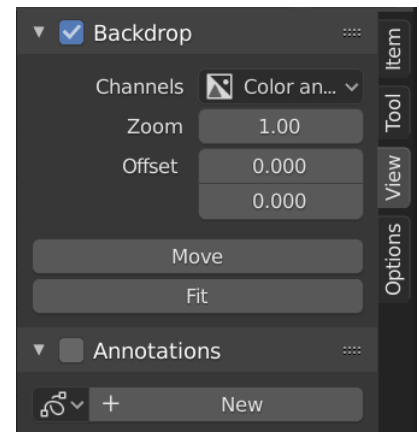
Change the screen space position of the backdrop.

Move Alt-MMB

Changes the position of the backdrop.

Fit

Scales the backdrop to fit the size of the Compositor.



Backdrop panel.

Reset Backdrop

Sets back to the default values of Zoom to 1 and Offset to 0.

Options

Reference

Panel:: *Sidebar region ▶ Options*

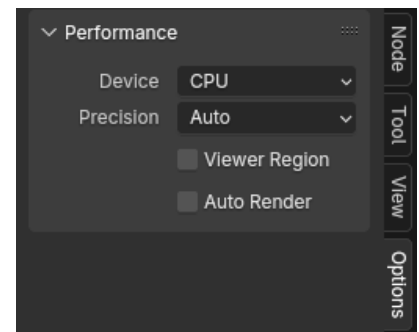
Performance

This panel helps you tweak the performance of the Compositor.

Device

The device used for compositing.

CPU::



Performance panel.

GPU:: Use the CPU for compositing.
Use the GPU for compositing.

Precision

The precision of compositor intermediate result.

Auto:: Use full precision for final renders, half precision otherwise.

Full:: Use full precision for final renders and viewport.

Viewer Region

This allows to set an area of interest for the backdrop. Press `Ctrl-B` and select a rectangular area in the preview which will become the next preview in the backdrop. `Ctrl-Alt-B` discards the region back to a full

preview. This is only a preview option, final compositing during a render ignores this region.

GPU Compositor

The new GPU accelerated compositor introduced in Blender 3.5 and is currently used for [viewport compositing](#).

Data

Dimensionality

Compositing nodes operate on data that is either an image or a dimensionless single value. For instance, the [Levels node](#) outputs a single value, while the [Render Layers node](#) outputs an image. Node inputs that expect a single value assume a default value if an image is given and ignore the image completely, for instance, the [Transform node](#) expects single values for its inputs and will assume default values if images were given to those inputs. The default values are those that are considered identity and thus have no effect on the output, so for the [Transform node](#), the *X*, *Y*, and *Angle* inputs will have a default value of zero, while the *Scale* input will have a default value of one. On the other hand, if node inputs that expect an image are given a single value, the single value will be assumed to cover the whole compositing space. For instance, the [Filter node](#) expect its *Factor* input to be an image, but if a single value is given, it will be assumed to be the same for all pixels.

Type

Three types of data exist, all of which are stored in half precision formats:

Float

A signed floating-point number. Integer data is also stored as floats because no integer type exist.

Vector

A 4D vector. While it is 4D, it can have different interpretations depending on the node that uses it. It can be treated as a 2D vector with the last two components ignored, for instance, the *Vector* input of the [Displace node](#) is treated as a 2D vector. It can be treated as a 3D vector with the last component ignored, for instance, the *Vector* input of the [Seperate XYZ node](#) is treated as a 3D vector. It can be treated as two consecutive 2D vectors. For instance the *Velocity Pass* as expected by the [Vector Blur node](#) is assumed to have the *2D Previous Velocity* in the X and Y components of the vector and the *2D Next Velocity* in the Z and W components of the vector.

Color

A 4D vector storing the Red, Green, Blue, and Alpha of the color. The color is free form and does not conform to a specific color space or alpha storage model, instead, appropriate nodes will have settings to control the representation of their output and nodes exist to convert between the different representations.

Implicit Conversion

In case a node input is given data of type other than its own type, the following implicit conversions are performed:

Source	Target	Conversion
--------	--------	------------

Float	Vector	$f \Rightarrow \text{Vector}(f, f, f, 0)$
-------	--------	---

Float	Color	$f \Rightarrow \text{Color}(f, f, f, 1)$
-------	-------	--

Vector	Float	$(x, y, z, w) \Rightarrow \text{Average}(x, y, z)$
--------	-------	--

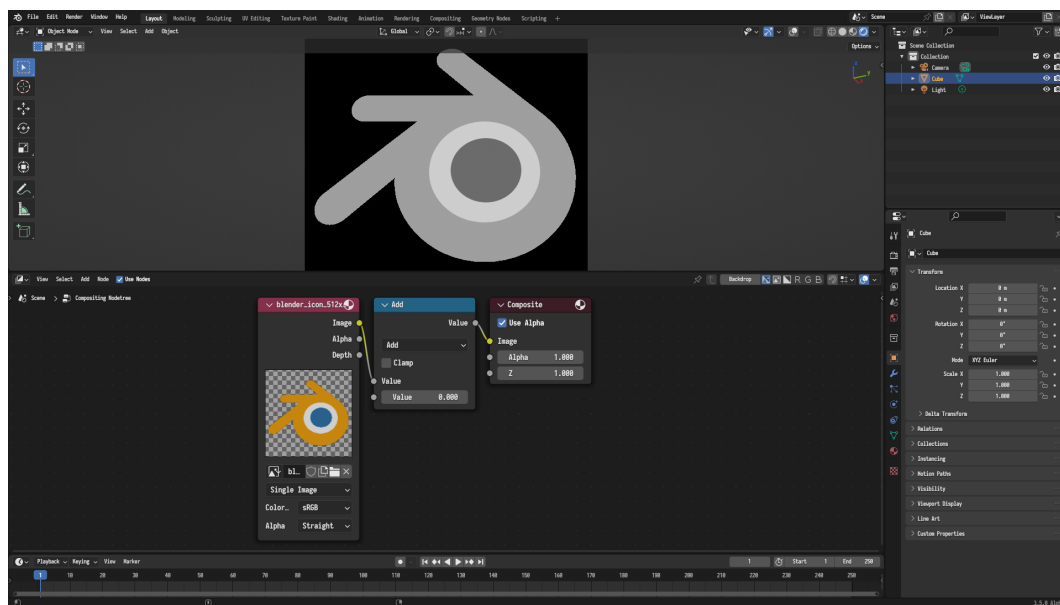
Vector	Color	$(x, y, z, w) \Rightarrow \text{Color}(x, y, z, 1)$
--------	-------	---

Source	Target	Conversion
100%	100%	100%
90%	90%	90%
80%	80%	80%
70%	70%	70%
60%	60%	60%
50%	50%	50%
40%	40%	40%
30%	30%	30%
20%	20%	20%
10%	10%	10%
0%	0%	0%

Color	Float	(r, g, b, a) => Average(r, g, b)
0	0.0	0.0
1	0.125	0.0625
2	0.25	0.125
3	0.375	0.1875
4	0.5	0.25
5	0.625	0.3125
6	0.75	0.375
7	0.875	0.4375
8	1.0	0.5
9	1.125	0.5625
10	1.25	0.625
11	1.375	0.6875
12	1.5	0.75
13	1.625	0.8125
14	1.75	0.875
15	1.875	0.9375
16	2.0	1.0

Color Vector (r, g, b, a) => Vector(r, g, b, 0)

The following example demonstrates implicit conversion between a color type and a float type, since the [Math node](#) expect float inputs.



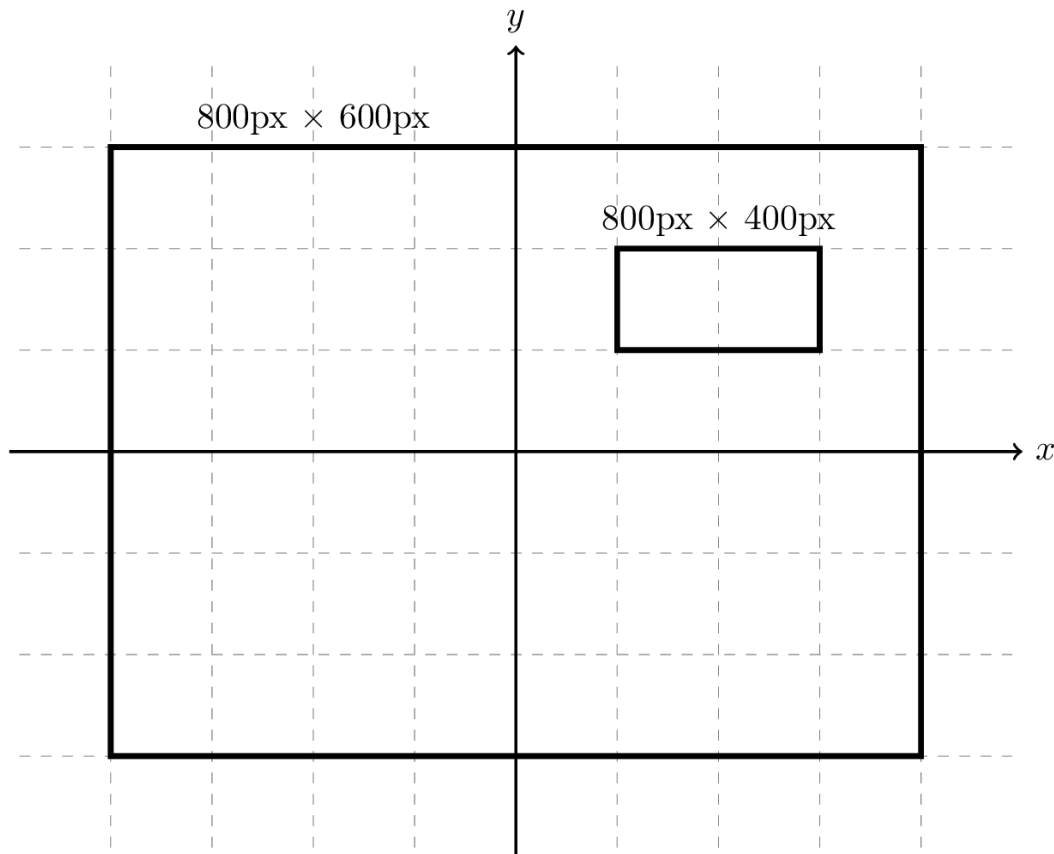
An example that demonstrates implicit conversion between a color type and a float type, since the *Math* node expects float inputs.

Compositing Space

Image Domain

The compositor is designed in such a way as to allow compositing in an infinite compositing space. Consequently, images are not only represented

by their size, but also by their transformation in that space, much like 3D objects have transformations. An identity transformation represents an image that is centered in space. The rectangular area occupied by an image in that space as defined by its transformation and size is called the *Domain* of the image. The figure below demonstrates the domains of two example images.



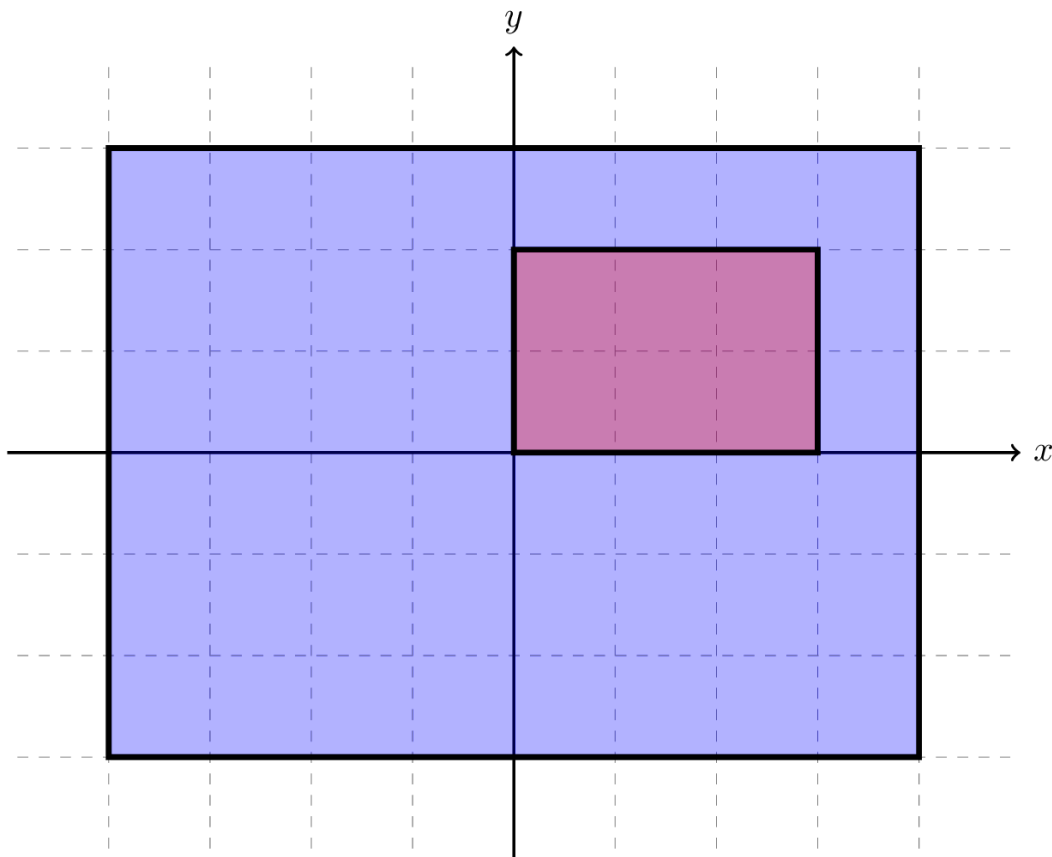
The domains of two example images are illustrated on the compositing space. One of the images is centered in space and the other is scaled down and translated such that it lies in the upper right quadrant of the space. Notice that both images have similar sizes in pixels, yet their apparent sizes are different.

Images can be transformed using nodes like the [Transform](#), [Translate](#), and [Rotate](#) nodes.

Operation Domain

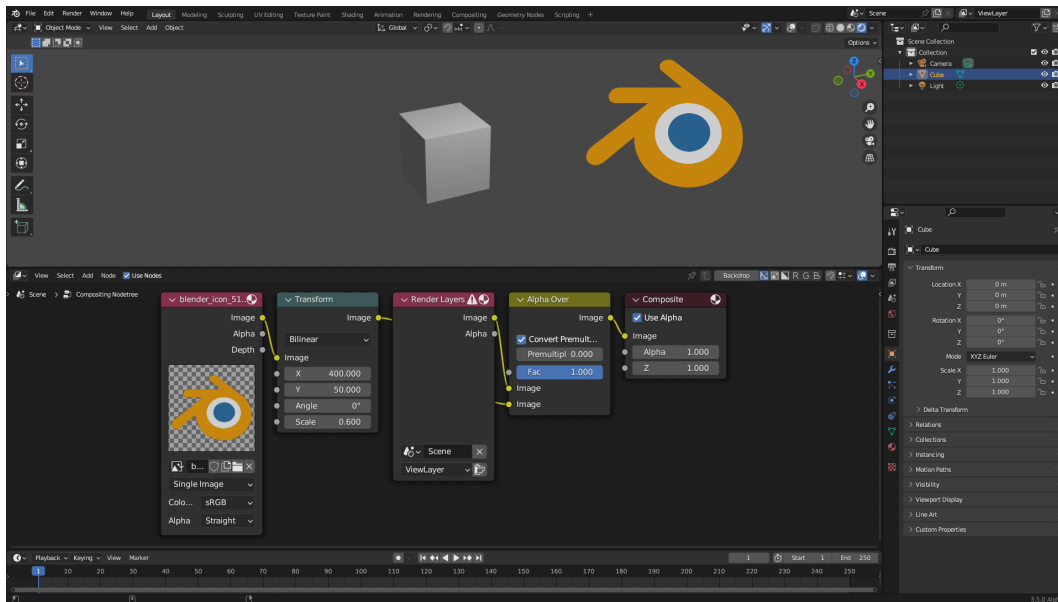
Compositor Nodes operate on a specific rectangular area of the compositing space called the *Operation Domain*. The nodes only consider the area of the input images that overlap the operation domain and ignores the rest of the images. If an input image doesn't completely overlap the operation domain, the rest of the operation domain for that input will be assumed to be a zero value, a zero vector, or a transparent zero color depending on the type.

For instance, the figure below illustrates a case where the operation domain of a node is the large blue area and the domain of an input image is the small red area. In that case, the input image doesn't completely overlap the operation domain, so the rest of the blue area for that input image is assumed to be zero.



An example case where the operation domain of a node is shown in blue and the domain of an input image is shown in red. Since the input image doesn't completely cover the operation domain of the node, the rest of the blue area for that input image is assumed to be zero.

The previous illustration is a representation of a real world example where one uses the [Alpha Over](#) node to overlay a small logo on an image, as shown in the figure below. In that case, the operation domain covers the entirety of the viewport — as will later be demonstrated, but the logo covers only a small area of it, so the rest of the area is assumed to be a zero transparent color, which is convenient for the use case.



A real world example where the Alpha Over node is used to overlay a small logo on an image. The logo only covers a small area of the operation domain, which is the whole viewport in this case, so the rest of the area is assumed to be a zero transparent color.

Interpolation

If an input image to a node is not perfectly aligned with the operation domain of the node or have a different size in pixels, the node would typically need to do a process called Interpolation, where the input image is read at the exact positions of the pixels of the operation domain. This can be done using different interpolation methods, including Nearest-Neighbor, Bilinear, and Bicubic interpolations. Those interpolation methods are demonstrated in the following [Wikipedia gallery](#). Transformation nodes like the [Transform](#) and [Rotate](#) nodes include an interpolation option to set how they prefer their output image to be read and interpolated.

Determining Operation Domain

The question remains on how nodes determine their operation domain. Different node types can have different mechanisms for determining their operation domain. But generally, three classes of nodes exist when it comes to the mechanism of determining the operation domain, each of which is presented in one of the following sections.

Input Nodes

The operation domain of input nodes like the [Image node](#) is a domain with an identity transformation and the same size as their outputs, so for the *Image* node, the operation domain will be the domain whose size is the size of the image and whose transformation is an identity one.

Output Nodes

The operation domain of output nodes like the [Viewer node](#) is a domain with an identity transformation and the same size as the final compositor output. For [viewport compositing](#), that size would be the viewport size, and for final render compositing, that size would be the scene render size.

Other Nodes

Unless stated otherwise in their respective documentation pages, all other nodes use the following mechanism. One of the inputs of the nodes is designated as the *Domain Input* of the node, and the operation domain of the node is identical to the domain of that designated input. For many nodes, the domain input can be intuitively identified as the main input of the node, for instance, the domain input for the [Filter node](#) is the *Image* input. But there are some caveats to note, which requires a deeper understanding of the mechanism.

Each input in the node has the so called *Domain Priority* property, the operation domain of the node is the domain of the non single value input with the highest domain priority. So for instance, the [Filter node](#) has two

inputs, the domain priority of the *Image* input is higher than that of the *Factor* input, and there are four possible configurations:

- Both the *Image* and factor inputs are connected to images. In this case, the *Image* input is the domain input because it has the highest priority and is connected to an image.
- The *Image* input is connected to an image, but the *Factor* input isn't. In this case, the *Image* input is the domain input because it is the only input connected to an image regardless of its priority.
- The *Image* input is not connected to an image but the *Factor* input is. In this case, the *Factor* input is the domain input because it is the only input connected to an image regardless of its priority.
- Neither the *Image* nor the *Factor* inputs are connected to images, in this case, there isn't a domain input because the node is evaluated on single values.

Considerations

The aforementioned mechanism for determining the operation domain has a number of consequences that needs to be considered as they might be undesirable, each of which is presented in one of the following sections.

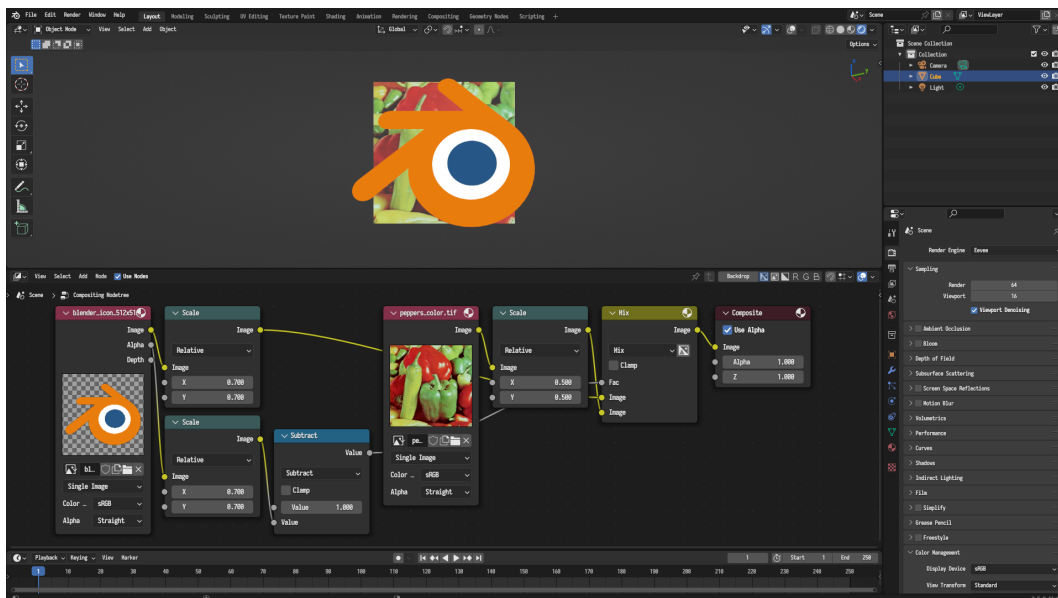
Clipping

The output of nodes will be intuitively clipped to the operation domain, or rather, the domain of the domain input. For instance, if the *Foreground* input is bigger than the *Background* input in the [Alpha Over node](#), the output will be clipped to the *Background* input because it is the domain input, as shown in the following figure.



The *Foreground* input is bigger than the *Background* input in the *Alpha Over* node, so the output is intuitively clipped to the *Background* input because it is the domain input.

The [Alpha Over node](#) currently does not support changing the domain priority for its inputs, so as a workaround, one can use a [Mix node](#) to achieved the desired behavior, noting that the first *Image* input in the *Mix* node has the highest domain priority, as shown in the following figure.



Working around the clipping behavior of the *Alpha Over* node using a *Mix* node, noting that the first *Image* input in the *Mix* node has the highest domain priority

Output

The GPU compositor only supports a single active output target, that is, only one of the [Composite nodes](#) or [Viewer nodes](#) in the node tree will be considered active and the rest will be ignored. In particular, the compositor searches the *Active Node Tree Context* and falls back to the *Root Node Tree Context* if no active output was found in the active node tree context. The active node tree context is the node tree of an expanded node group, that is, when the users selects a node group node and edits its underlying tree, while the root node tree context is the top level node tree without any expanded node groups. The compositor searches for the active *Composite* node, if non was found, it searches for the active *Viewer* node, be it a *Viewer* or a *Split Viewer* node, if non was found, the compositor doesn't run altogether. Consequently, note that adding a *Viewer* node will have no effect if there is a *Composite* node, since the priority is given to *Composite* nodes.

Input Nodes

Input nodes produce information from a data source. For instance, an input can be:

- Taken directly from the active camera in a selected scene.
- A static image.
- A movie clip (such as an image sequence or video).
- A color or value.

These nodes generate the information that is passed to other nodes. As such, they have no input sockets; only outputs.

- [Constant](#)

-
- [Bokeh Image Node](#)
 - [Image Node](#)
 - [Mask Node](#)
 - [Movie Clip Node](#)
 - [Texture Node](#)

-
- [Scene](#)

Constant

- [RGB Node](#)
- [Value Node](#)

RGB Node



Inputs

This node has no input sockets.

Properties

The RGB node uses the [color picker widget](#).

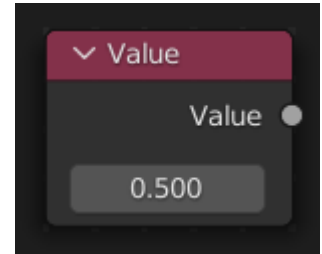
Outputs

Color / RGBA

A single RGBA color value.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.



Inputs

This node has no input sockets.

Properties

Single numerical value (floating-point).

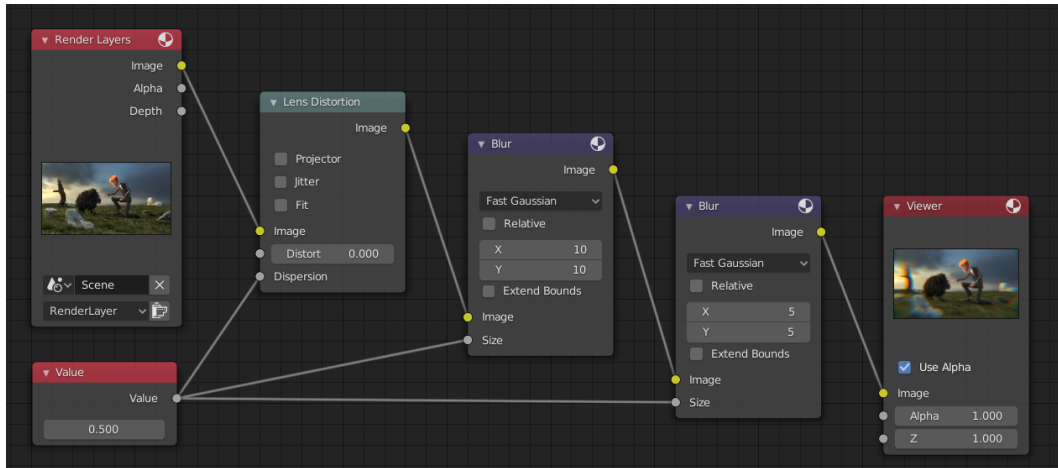
Outputs

Value

The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.



Example of the *Value* node.

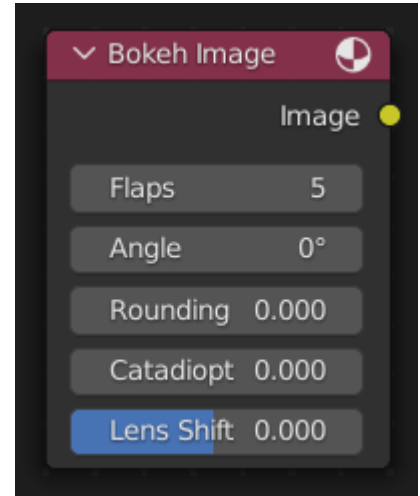
Tip

From this you can also make different values proportional to each other by adding a [Math Node](#) in between the different links.

Bokeh Image Node

The *Bokeh Image* node generates a special input image for use with the [Bokeh Blur](#) filter node.

The *Bokeh Image* node is designed to create a reference image which simulates optical parameters such as aperture shape and lens distortions which have important impacts on bokeh in real cameras.



Inputs

This node has no input sockets.

Properties

The first three settings simulate the aperture of the camera.

Flaps

Sets an integer number of blades for the cameras iris diaphragm.

Angle

Gives these blades an angular offset relative to the image plane.

Rounding

Sets the curvature of the blades with (0 to 1) from straight to bringing them to a perfect circle.

Catadioptric

Provides a type of distortion found in mirror lenses and some telescopes. This can be useful to produce a visual complex bokeh.

Lens Shift

Introduces chromatic aberration into the blur such as would be caused by a tilt-shift lens.

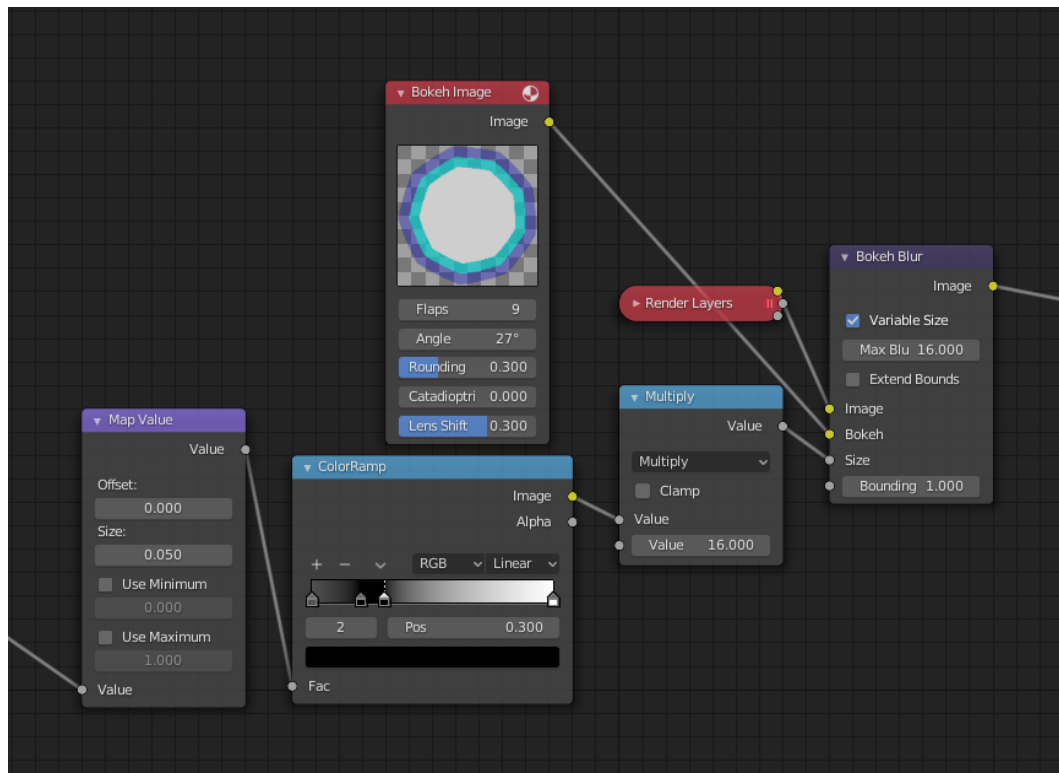
Outputs

Image

The generated bokeh image.

Example

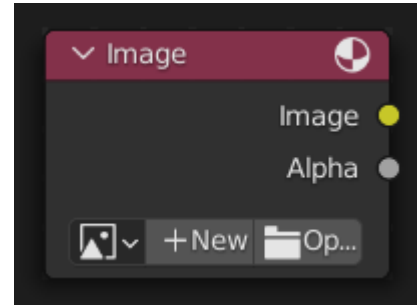
In the example below the *Bokeh Image* is used to define the shape of the bokeh for the [Bokeh Blur](#) node.



Example of *Bokeh Image* node.

Image Node

The *Image* node injects any image [format that is supported by Blender](#).



Inputs

This node has no input sockets.

Properties

Image

Selection of different types of media. For controls see [Data-Block Menu](#). For the options see [Image Settings](#).

Note

More options can be set in the Sidebar region.

Outputs

The first two sockets are the minimum.

Image

Standard color output.

Alpha

Separate Alpha value.

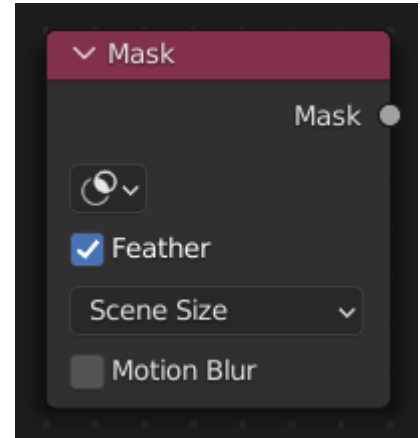
Note

Multi-Layer Format

When a multi-layer file format, like EXR, is loaded, each layer is made available as a socket.

Mask Node

The Mask node can be used to select a [Mask data-block](#). This node can be used with other nodes, for example to Invert, Multiply or Mix, or used as a factor input.



Inputs

This node has no input sockets.

Properties

Masks

The selectable mask data-block. If the label is left blank, the mask name will be set.

Feather

Use or ignore feather points defined for splines see [Mask Feathers](#) for more details.

Size

Scene Size will give an image the size of the render resolution for the scene, scaling along when rendering with different resolutions. Fixed gives a fixed size in pixels. Fixed/Scene gives a size in pixels that still scales along when changing the render resolution percentage in the scene.

Motion Blur

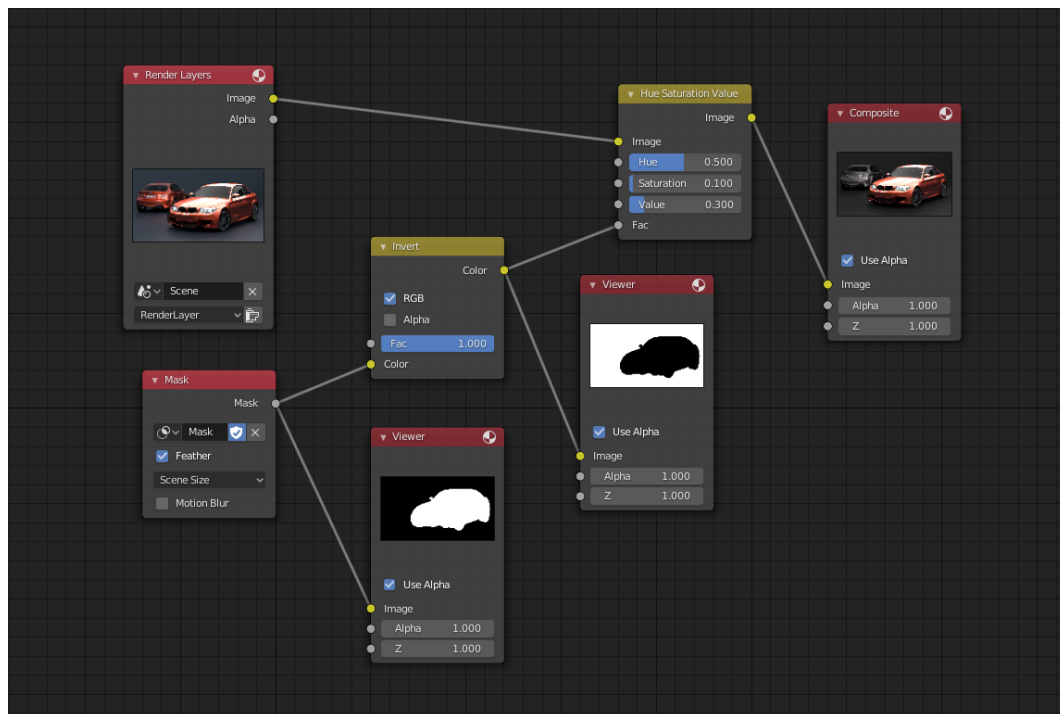
For animated masks, creating a motion blurred mask from the surrounding frames, with a given number of samples (higher gives better quality), and a camera shutter time in seconds.

Outputs

Mask

The black-and-white output of the mask.

Example

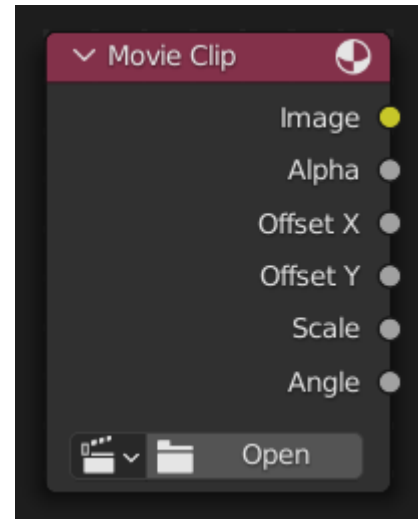


Example of the Mask node.

In the example above, the *Mask* node is used to isolate the object from the background to preserve it from being corrected.

Movie Clip Node

This node is a special node that uses some of the values taken from footage cameras and trackings and link them to the output. It is possible to load image sequences, but only Image and Alpha values will be available, because the other outputs will not have any values associated with them. When a tracked clip is chosen, Blender will fulfill the outputs using internal values taken from the tracking. So the controls for start and end frames will be defined in the Movie Clip editor.



Inputs

This node has no input sockets.

Properties

Movie Clip

Used to select the movie clip. For controls see [Data-Block Menu](#).

Outputs

The first two sockets are the minimum output.

Image

Outputs the entire image in the specified color space.

Alpha

The alpha value taken from the movie or image.

Offset X

The X offset value from the footage camera or tracking.

Offset Y

The Y offset value from the footage camera or tracking.

Scale

The scale of the image taken from the footage camera or tracking.

Angle

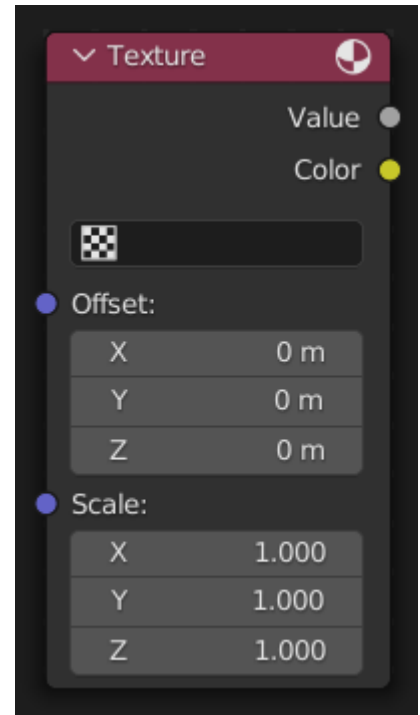
The lens angle taken from the footage camera or tracking.

Texture Node

The Texture node makes 3D textures available to the Compositor.

Note

The Texture node is not supported by the [viewport compositor](#).



Inputs

Offset

A vector (XYZ) transforming the origin of the texture.

Scale

A vector (XYZ) to scale the texture.

Properties

Texture

The texture can be selected from a list of textures available in the current blend-file or linked-in textures. The textures themselves can not be edited in the Compositor, but in the [Texture Node Editor](#).

Outputs

Value

Gray-scale color values.

Color

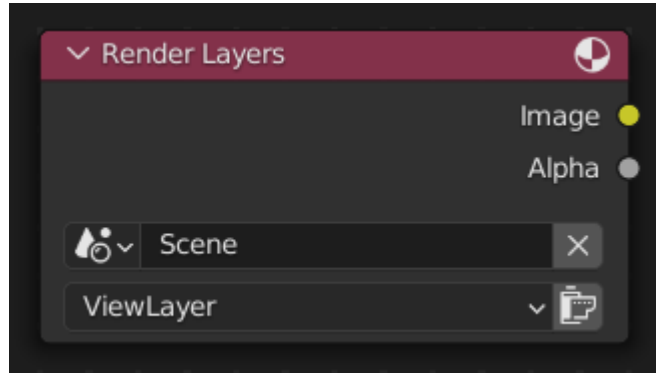
Color values.

Scene

- [Render Layers Node](#)
- [Scene Time Node](#)
- [Time Curve Node](#)

Render Layers Node

Read render layers and passes from a scene into the compositing node graph.



Inputs

This node has no input sockets.

Properties

Scene

Select the scene within your blend-file. The scene information taken is the raw footage (pre-compositing and pre-sequencing).

Hint

To use composited footage from another scene, it has to be rendered into a multi-layer frameset (e.g. OpenEXR) as an intermediate file store and then imported with Image input node again.

Render Layer

A list of available [Render Layers](#). The render button is a shorthand to re-render the active scene.

Outputs

Image

Rendered image.

Alpha

Alpha channel.

Render Passes Sockets

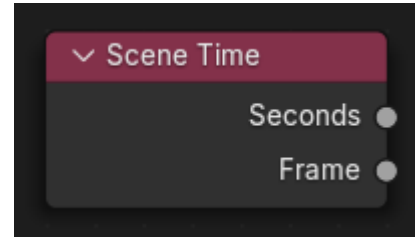
Depending on the Render passes that are enabled, other sockets are available. See [render passes](#).

Note

In the viewport compositor, Render Passes are only supported in EEVEE. For other engines, the passes return a zero value, a zero vector, or a transparent color depending on their type.

Scene Time Node

The *Scene Time* node outputs the current time in the scene's animation in units of seconds or frames.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Seconds

Current scene time in seconds.

Frames

Current scene frame. As an input in geometry nodes, this output may also output non-round numbers, in order to support higher quality [motion blur](#).

Time Curve Node

The *Time Curve node* generates a factor value (from 0.0 to 1.0) between the scene start and end time, using a curve mapping.



Inputs

This node has no inputs.

Properties

Curve

The Y value defined by the curve is the factor output. For the curve controls, see [Curve widget](#).

Tip

Flipping the curve around reverses the time input, but doing so is easily overlooked in the node setup.

Start, End

Start frame and End frame of the range of time specifying the values the output should last. This range becomes the X axis of the graph. The time input could be reversed by specifying a start frame greater than the end frame.

Outputs

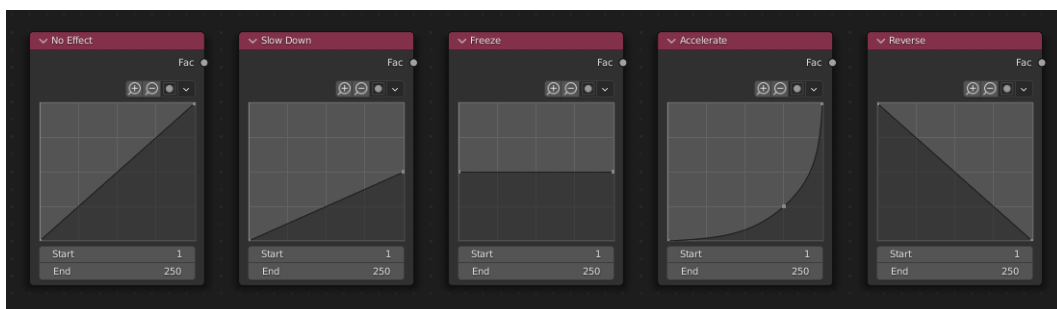
Factor

The Y value of the curve at the current frame.

Hint

The [Map Value](#) node can be used to map the output to a more appropriate value. With some curves, it is possible that the Time Curve node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

Example



Time controls from left to right: no effect, slow down, freeze, accelerate, reverse.

Output Nodes

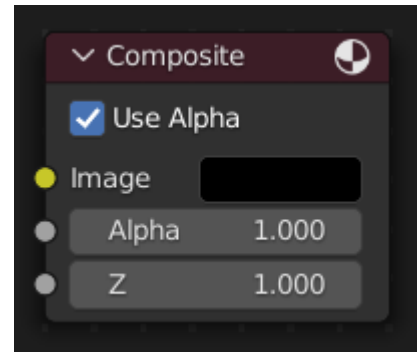
These nodes are used to output the composited result in some way.

- [Composite Node](#)
- [Viewer Node](#)

-
- [File Output Node](#)

Composite Node

The Composite node is where the actual output from the Compositor is connected to the renderer. This node is updated after each render, but also reflects changes in the node tree (provided at least one finished input node is connected).



Inputs

Connecting a node to the Composite node will output the result of the prior tree of that node to the Compositor.

Image

RGB image. The default is black, so leaving this node unconnected will result in a black image.

Alpha

Alpha channel.

Properties

Use Alpha

Used alpha channel, colors are treated alpha *premultiplied*. If disabled, alpha channel gets set to 1, and colors are treated as alpha *straight*, i.e. color channels does not change.

Outputs

This node has no output sockets.

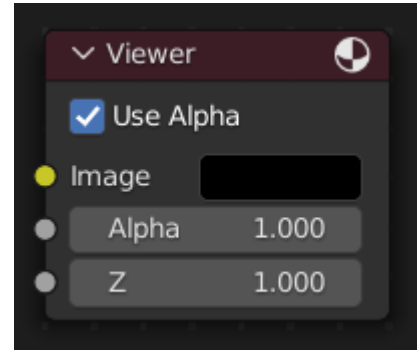
Note

If multiple Composite nodes are added, only the active one (last selected, indicated by a red header) will be used.

Viewer Node

The *Viewer* node allows temporarily visualizing data from inside a node graph. It can be plugged in anywhere to inspect an image or value map in your node tree.

Select a view node with LMB to switch between multiple viewer nodes. It is possible to automatically plug any other node into a Viewer node by pressing Shift-Ctrl-LMB on it.



Inputs

Image

RGB image. The default is black, so leaving this node unconnected will result in a black image.

Alpha

Alpha channel.

Properties

Use Alpha

Used alpha channel, colors are treated alpha *premultiplied*. If disabled, alpha channel gets set to 1, and colors are treated as alpha *straight*, i.e. color channels does not change.

Outputs

This node has no output sockets.

Note

It is possible to add multiple Viewer nodes, though only the active one (last selected, indicated by a red header) will be shown on the backdrop or in the Image editor.

Using the Image Editor

The Viewer node allows results to be displayed in the Image Editor. The image is facilitated in the header by selecting *Viewer Node* in the linked *Image* data-block menu. The Image Editor will display the image from the currently selected Viewer node.

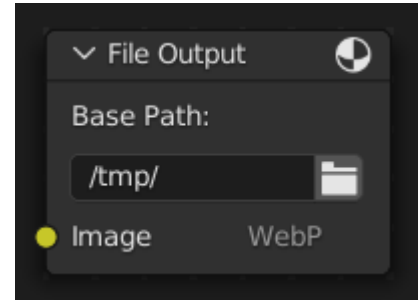
To save the image being viewed, use *Image ▶ Save As...*, Alt-S to save the image to a file.

The Image Editor also has three additional options in its header to view Images with or without Alpha, or to view the Alpha or Z itself. Click and holding the mouse in the Image displayed allows you to sample the values.

File Output Node

This node writes out an image, for each frame range specified, to the filename entered, as part of a frameset sequence.

This node can be used as a way to automatically save the image after a render; In addition, since this node can be hooked in anywhere in the node tree, it can also save intermediate images automatically.



Inputs

Image

The image(s) will be saved on rendering, writing to the current frame. An entire sequence of images will be saved, when an animation is rendered.

Properties

Base Path

Unlike the render output filepath, this node uses a base directory and an image name, by default the output path is composed of: {base path}/{file name}{frame number}.{extension}.

Besides being split into two settings, in all other respects, this setting is treated the same as the [render output path](#).

File Format

Label that shows the selected file format.

Note

More options can be set in the Sidebar region.

Outputs

This node has no output sockets.

Color Nodes

These nodes adjust the image's colors, for example increasing the contrast, making it warmer, overlaying another image, etc.

- [Adjust](#)
-

- [Mix](#)
-

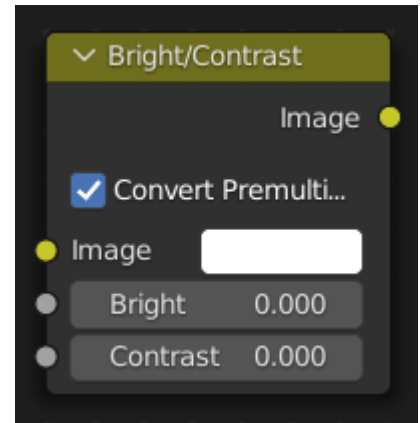
- [Alpha Convert Node](#)
 - [Color Ramp Node](#)
 - [Convert Colorspace Node](#)
 - [Set Alpha Node](#)
-

- [Invert Color Node](#)
- [RGB to BW Node](#)

Adjust

- [Brightness/Contrast Node](#)
- [Color Balance Node](#)
- [Color Correction Node](#)
- [Exposure Node](#)
- [Gamma Node](#)
- [Hue Correct Node](#)
- [Hue/Saturation/Value Node](#)
- [RGB Curves Node](#)
- [Tone Map Node](#)

Brightness/Contrast Node



Inputs

Image

Standard color input.

Brightness

An additive-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast

A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

Convert Premultiplied

By default, it is supposed to work in *premultiplied* alpha. If the *Convert Premul* checkbox is not enabled, it is supposed to work in *straight* alpha.

See [Alpha Channel](#).

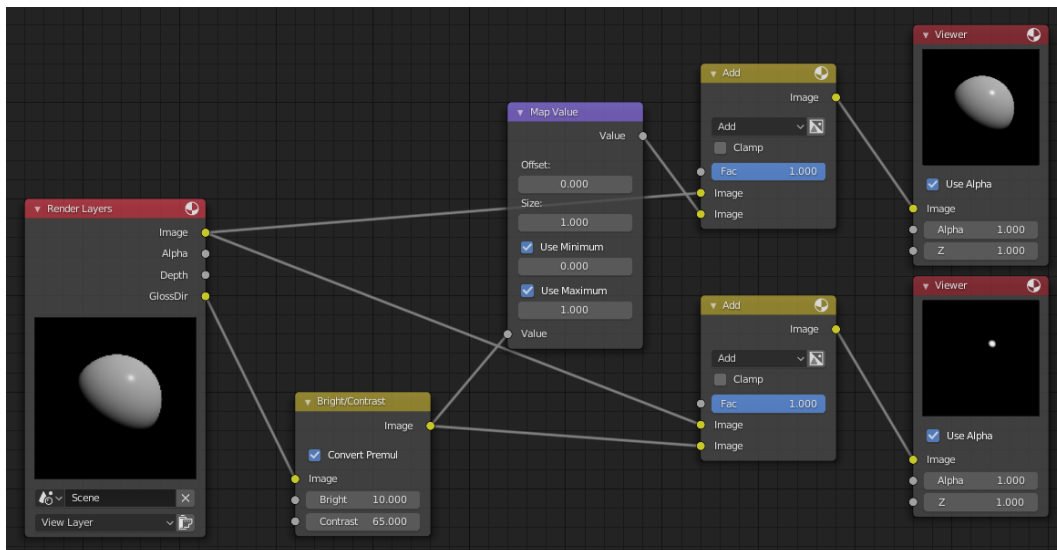
Outputs

Image

Standard color output.

Notes

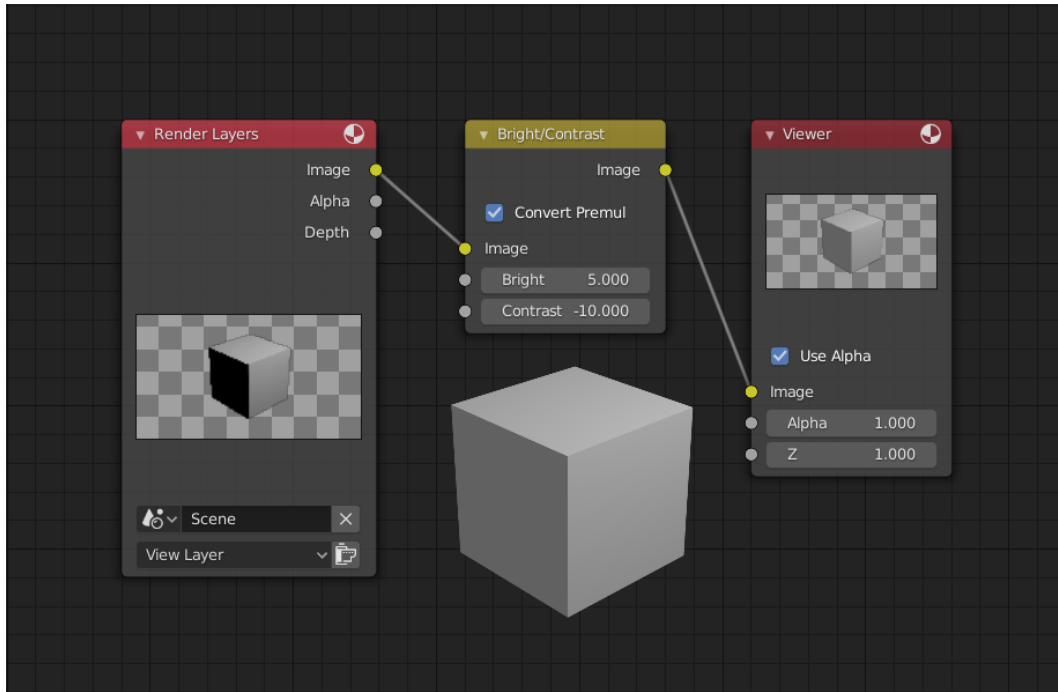
It is possible that this node will put out a value set that has values beyond the normal range, i.e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).



Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to intensify the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has a value much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

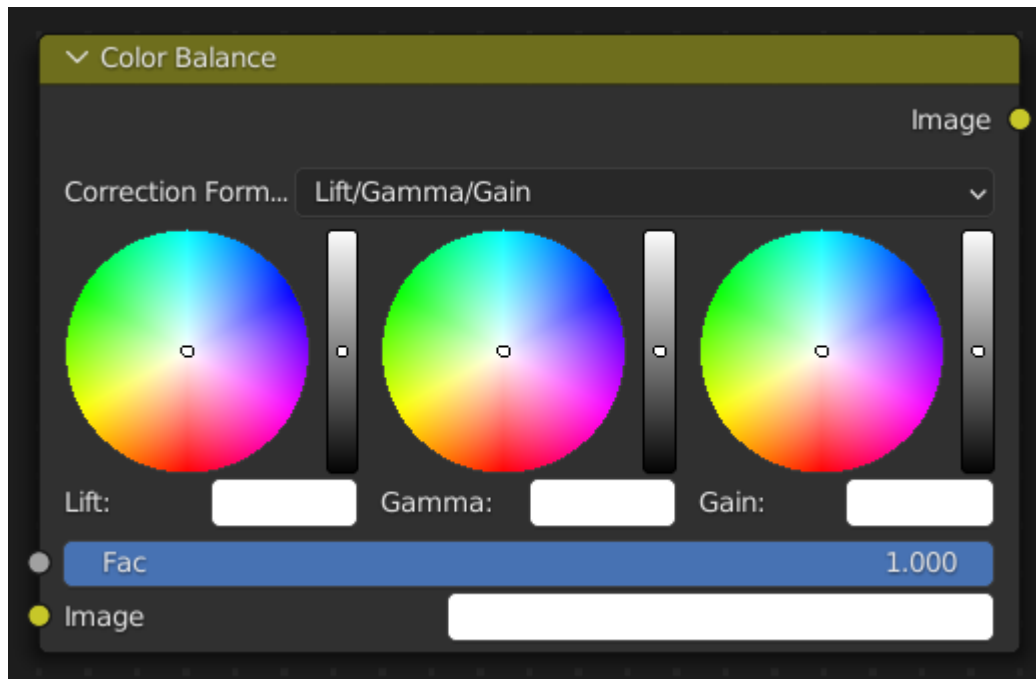
Example



A basic example.

Color Balance Node

The Color Balance node adjusts the color and values of an image.



Inputs

Factor

Controls the amount of influence the node exerts on the output image.

Color

Standard color input.

Properties

Correction Formula

The mathematical method to adjust the image's colors.

Lift/Gamma/Gain:: Adjusts the colors and tonal range of an image by controlling the shadows, midtones, and highlights separately.

Lift

Adjusts the darkest areas of the image (the shadows).

Gamma

Primarily affects the midtones, the middle range of brightness in the image.

Gain

Controls the brightest parts of the image (the highlights).

Offset/Power/Slope (ASC-CDL)::

A standardized model for adjusting the colors and tonal range of an image. This allows the same values to be used across different application to yield the same result. See [Advanced](#) for more details on the underlying implementation.

Offset

Adjusts the darkest areas of the image (the shadows).

Basis

Additional offset, allows to specify a negative offset value.

Power

Primarily affects the midtones, the middle range of brightness in the image.

Slope

Controls the brightest parts of the image (the highlights).

White Point::

Adjusts the color that should be considered white. The white point is specified as setting the inputs color temperature and then the desired output temperature.

Temperature

The blackbody temperature of the primary illuminant. By default a D65 white point is used.

Tint

The amount of green/magenta shift of the blackbody curve.

Outputs

Color

Standard output image.

Advanced

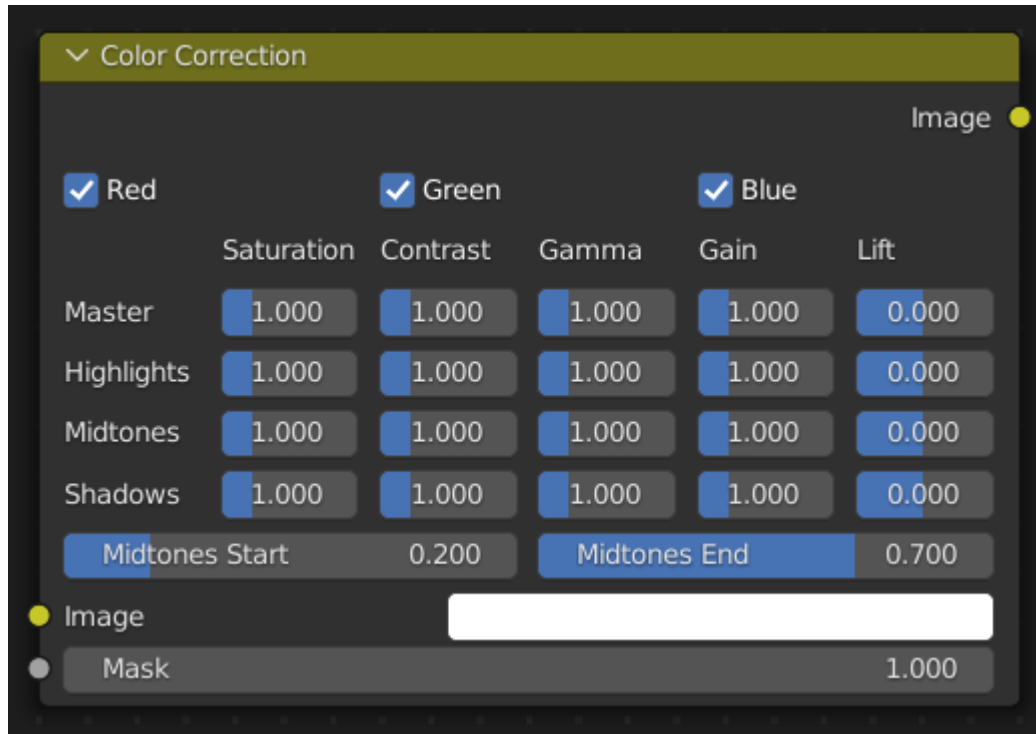
The Offset/Power/Slope Formula

$$\text{out} = (i \times s + o)^p$$

where:

- *out*: The color graded pixel code value.
- *i*: The input pixel code value (0 to 1) (black to white).
- *s*: Slope (any number 0 or greater, nominal value is 1.0).
- *o*: Offset (any number, the nominal value is 0).
- *p*: Power (any number greater than 0, nominal value is 1.0).

Color Correction Node



The Color Correction node adjusts the color of an image, separately in several tonal ranges (highlights, midtones and shadows).

Properties

Red, Green, Blue

Specifies which RGB channels will be affected by the correction.

Correction Tools (Columns)

Saturation

Adjusts the image's saturation.

Contrast

Adjust image contrast.

Gamma

Exponential gamma correction, affecting the midtones of the image.
(Works like Power in the Color Balance node.)

Gain

Multiplier, stronger influence on the highlights. (Works like Slope in the Color Balance node.)

Lift

This value (can be negative) will be added (+), linear lightens or darkens the image. (Works like *Offset* in the Color Balance node.)

Tonal Ranges (Rows)

Master

These sliders affect the entire tonal range.

Highlights

These sliders only affect the highlights.

Midtones

These sliders only affect the midtones.

Shadows

Affects the dark tones of an image often affecting the shadows.

Midtones Start, Midtones End

Defines the start and the end of midtones range, i.e. values where the whole tonal range is divided into the highlights, midtones and shadows (there is also a smooth transition between the ranges of width 0.2 units).

Inputs

Image

Standard color input.

Mask

Controls the amount of influence the node exerts on the output image.

Outputs

Color

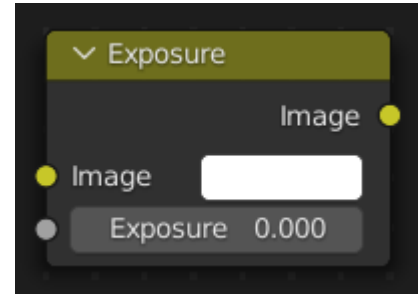
Standard color output.

Exposure Node

The Exposure Node adjusts the brightness of an image using a camera exposure parameter.

See also

The exposure can also be adjusted in the scene [Color Management](#).



Inputs

Image

Standard color input.

Exposure

Scalar factor to adjust the exposure of the image.

Properties

This node has no properties.

Outputs

Image

Standard color output.

Examples

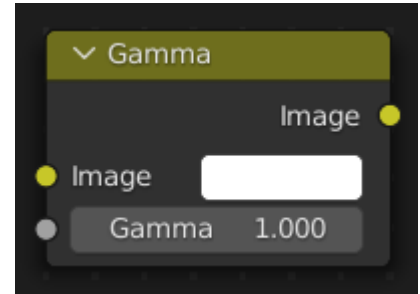
In the example below, the Exposure node is used to increase the brightness of the window area using a mask.



Example of an Exposure node.

Gamma Node

Use this node to apply a gamma correction.



Inputs

Image

Standard color input.

Gamma

An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image

Standard color output.

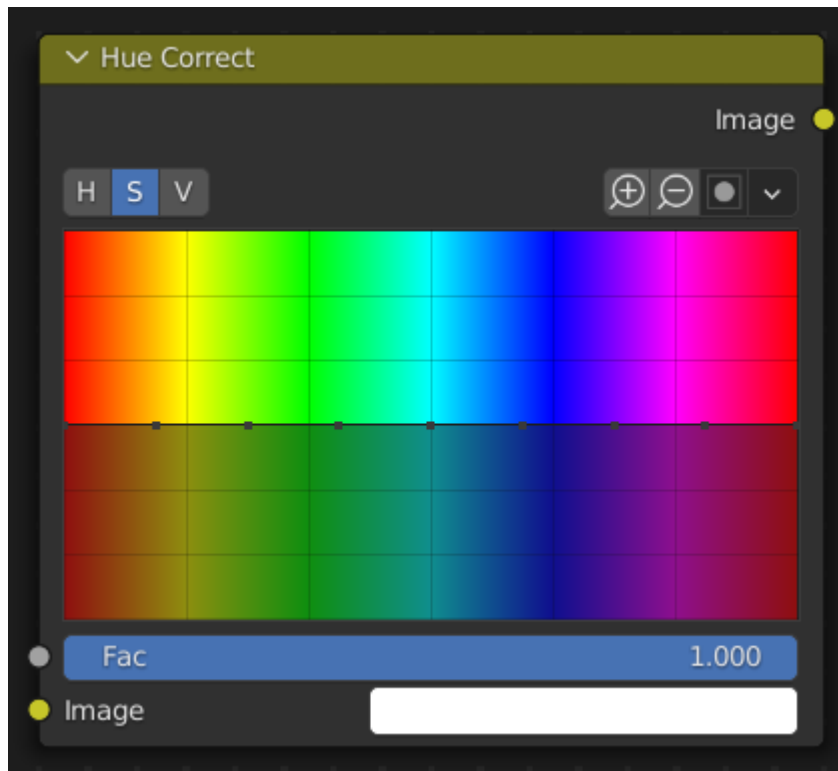
Examples



Example of a Gamma node.

Hue Correct Node

The *Hue Correct Node* adjusts the hue, saturation, and value of an image, with an input curve.



Inputs

Factor

Controls the amount of influence the node exerts on the output image.

Image

Standard color input.

Properties

Level

H (Hue), S (Saturation), V (Value)

Curve

For the curve controls see: [Curve widget](#). By default, the curve is a straight line, meaning there is no change. The spectrum allows you to raise or lower HSV levels for each range of pixel colors. To change an H, S, or V level, move the curve points up or down. Pixels with hue values each point in the horizontal position of the graph will be changed depending on the shape of the curve.

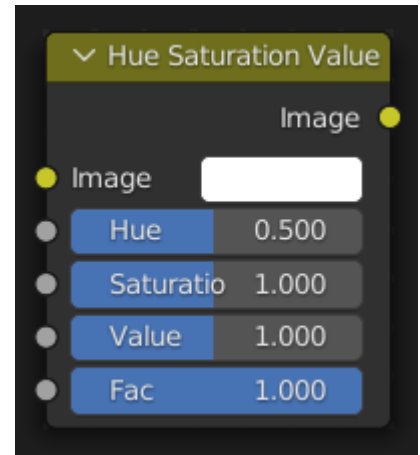
Outputs

Image

Standard color output.

Hue/Saturation/Value Node

The *Hue/Saturation/Value Node* applies a color transformation in the HSV [Color Model](#).



Inputs

Factor

The amount of influence the node exerts on the image.

Image/Color

Standard color input.

Hue

The hue rotation offset, from 0 (-180°) to 1 (+180°). Note that 0 and 1 have the same result.

Saturation

A value of 0 removes color from the image, making it black-and-white. A value greater than 1.0 increases saturation.

Value

The value shift. 0 makes the color black, 1 keeps it the same, and higher values make it brighter.

Outputs

Image/Color

Standard color output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are laid out on a circle

If you apply a Hue offset of 1 (+180°) to a blue image, you get the diametrically opposite color, which is yellow. If you apply a Hue offset of 1 to that yellow image, you get blue again.

Grayscale images have no hue

Trying to change the Hue or Saturation of a grayscale image has no effect. You can only brighten or darken it by adjusting the Value. To add color, use the Mix node instead.

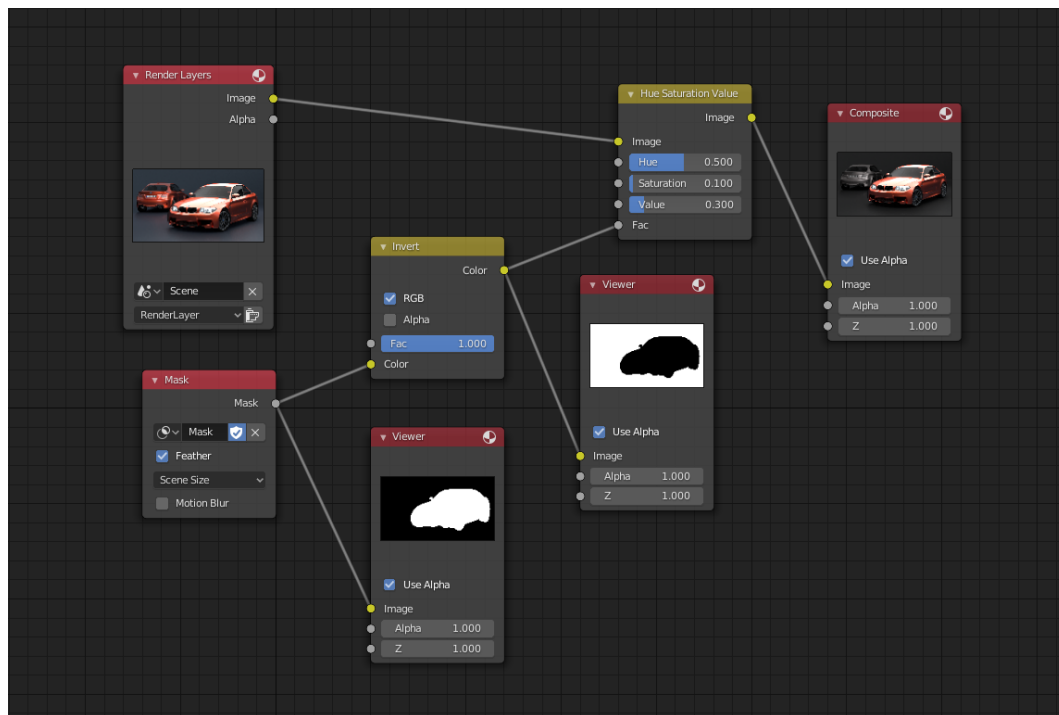
Changing the effect over time

The different values can be animated using a *Time Curve* node or by setting keyframes.

HSV Example



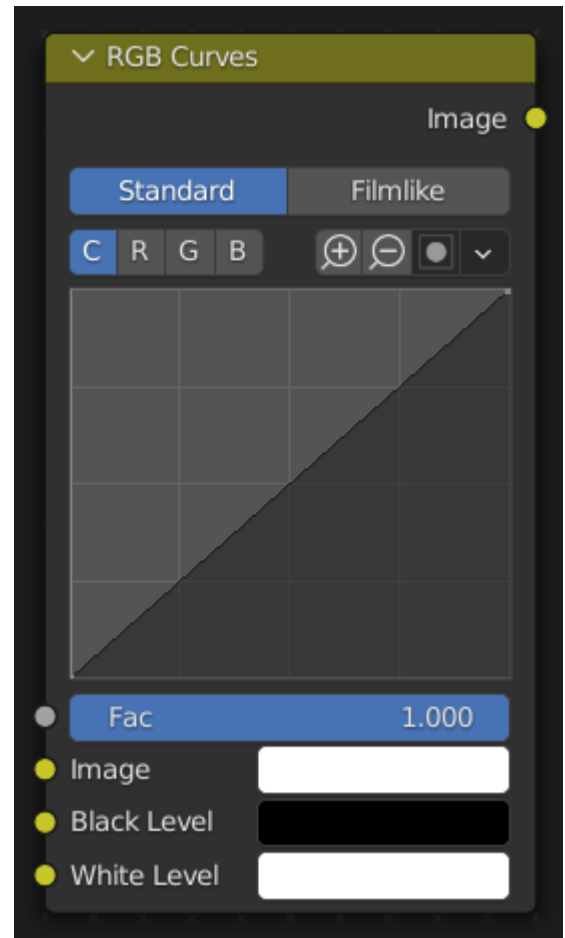
A basic example.



An example of using the Factor input for masking.

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.



Inputs

Factor

Controls the amount of influence the node exerts on the image.

Image/Color

Standard color input.

Black Level *Compositor Only*

Defines the input color that should be mapped to black.

White Level *Compositor Only*

Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the [eyedropper](#) to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel

The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve

A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see [Curve widget](#).

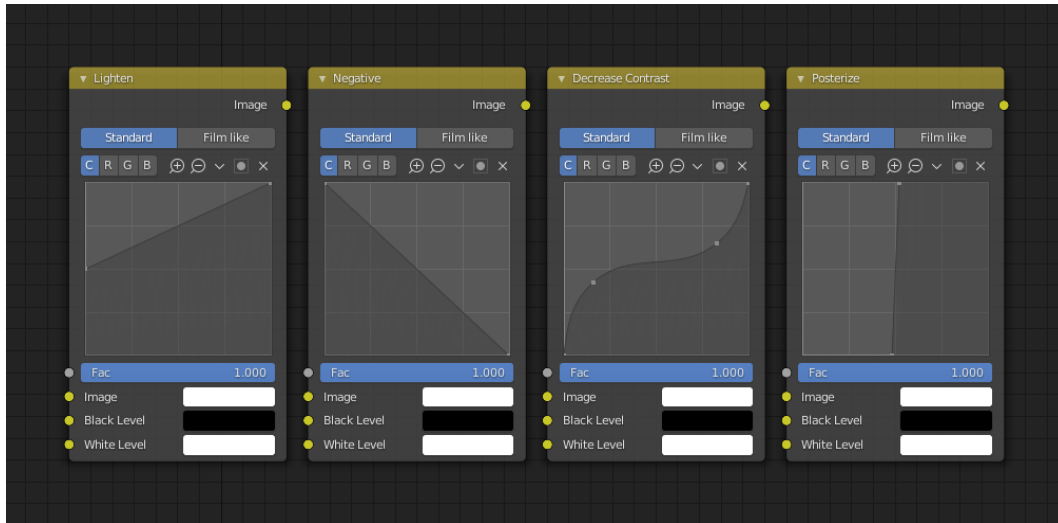
Outputs

Image/Color

Standard color output.

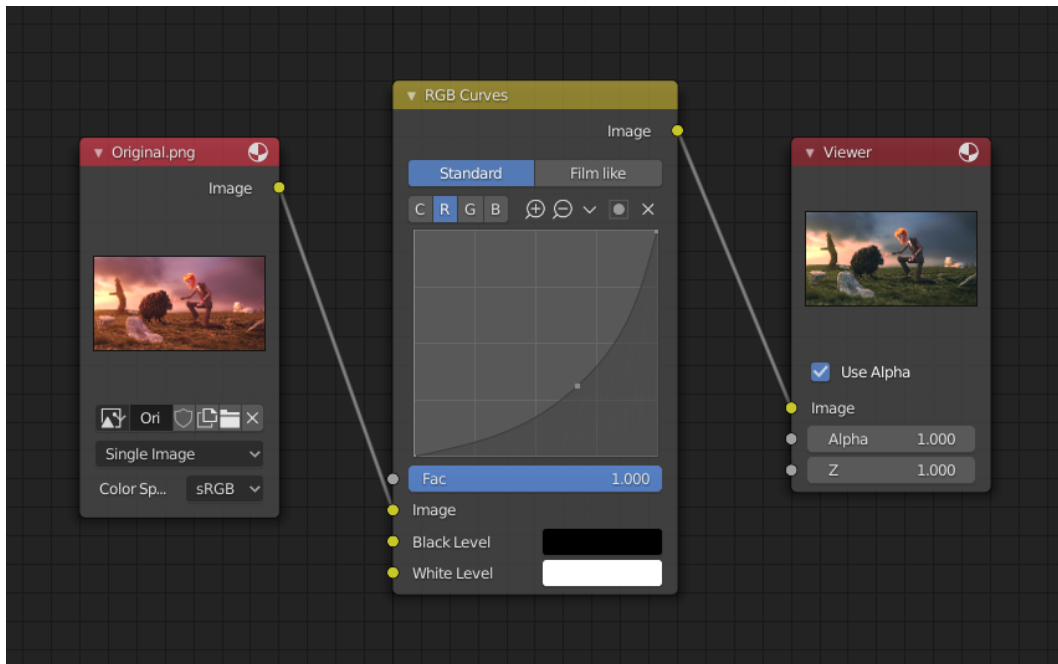
Examples

Below are some common curves you can use to achieve desired effects.



From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

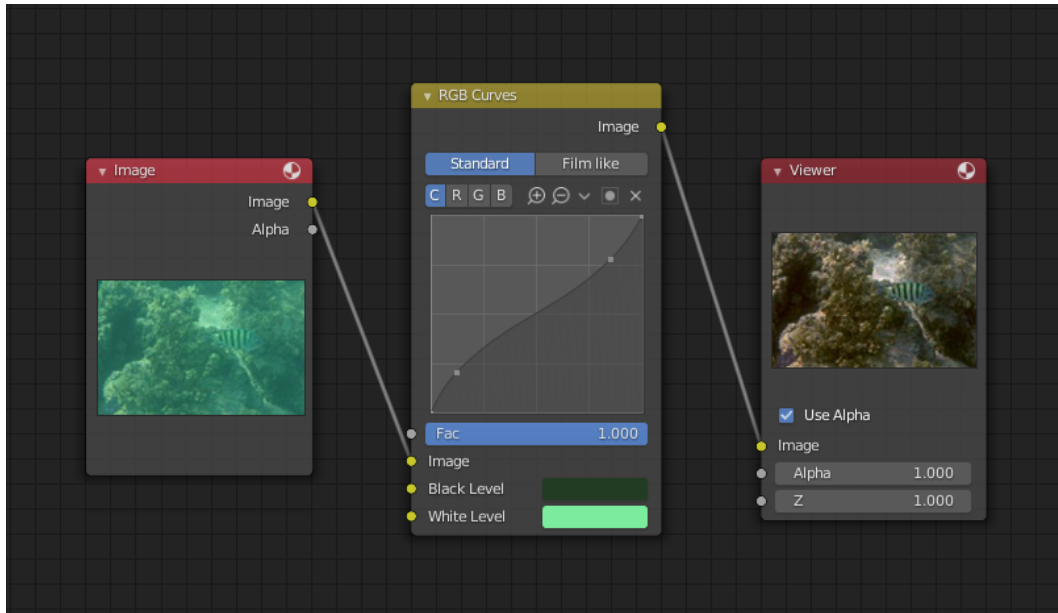


Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the [Mix Node](#) has an additional example about fixing overexposure.

Color Correction using Black/White Levels



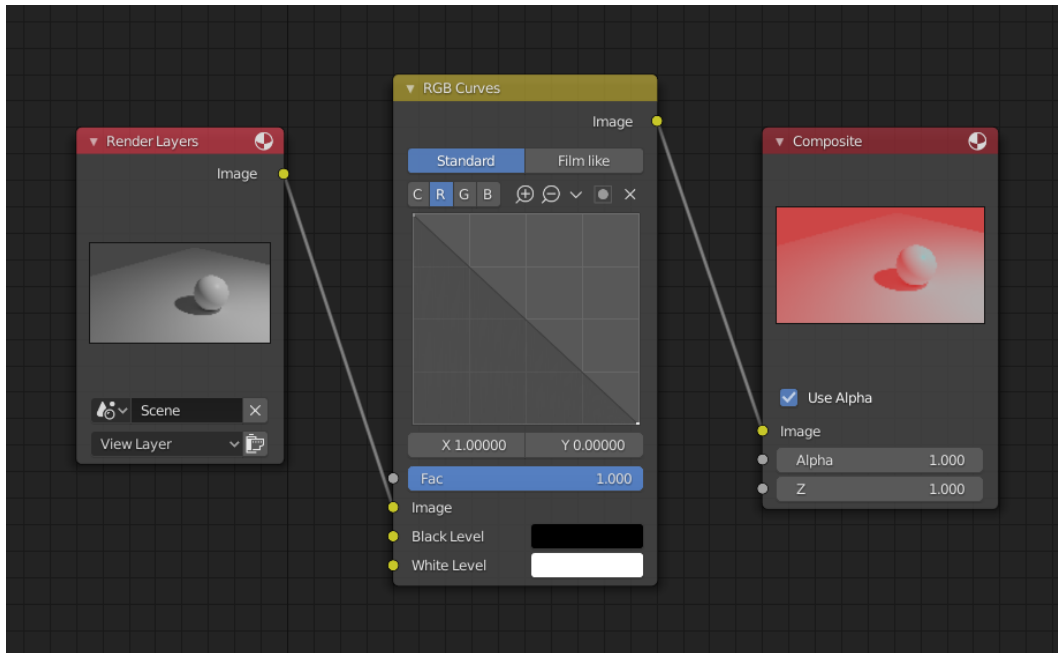
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

Effects

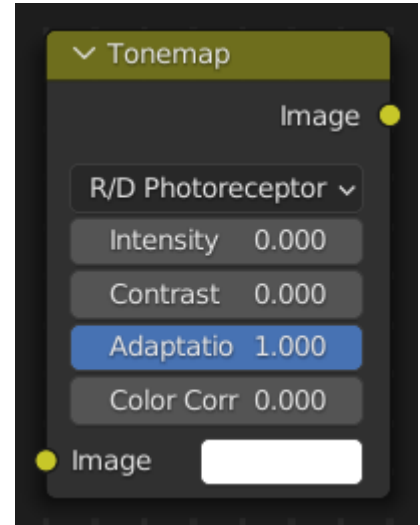


Changing colors by inverting the red channel.

Tone Map Node

Tone mapping is used to map high dynamic range colors into a more limited dynamic range supported by the display, while preserving the appearance as much as possible.

This is a legacy node. It is recommended to use view transforms in the color management settings instead, and output linear high dynamic range images from the compositor instead of low dynamic range.



Inputs

Image
HDR image.

Properties

Type

Rh Simple

Key

The value the average luminance is mapped to.

Offset

Normally always 1, but can be used as an extra control to alter the brightness curve.

Gamma

If not used, set to 1.

R/D Photoreceptor

Intensity

If less than zero, darkens image; otherwise, makes it brighter.

Contrast

Set to 0 to use estimate from input image.

Adaptation

If 0, global; if 1, based on pixel intensity.

Color Correction

If 0, same for all channels; if 1, each independent.

Outputs

Image

LDR image.

Mix

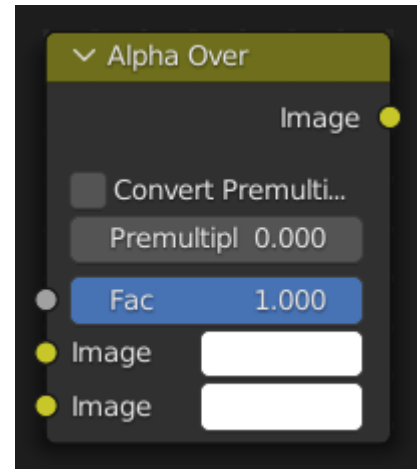
- [Alpha Over Node](#)

-
- [Combine Color Node](#)
 - [Separate Color Node](#)

-
- [Mix Color](#)
 - [Z Combine Node](#)

Alpha Over Node

The *Alpha Over* node is used to layer an image on top of another with alpha blending.



Inputs

Factor

The alpha of the foreground image, going from 0 (fully transparent) to 1 (fully opaque).

Image

The background image.

Image

The foreground image.

Properties

Convert Premultiplied

The *Alpha Over* node expects the foreground image to use [Premultiplied Alpha](#). If it uses [Straight Alpha](#) instead, you can enable this checkbox to convert it.

Premultiplied

Interpolate between [Premultiplied Alpha](#) and [Straight Alpha](#).

When set to 1, the foreground color values will be multiplied by the alpha; this is equivalent to enabling *Convert Premultiplied*. When set to 0, the color values do not change.

If *Premultiplied* is not zero, *Convert Premultiplied* will be ignored.

Note

This is a legacy option.

Outputs

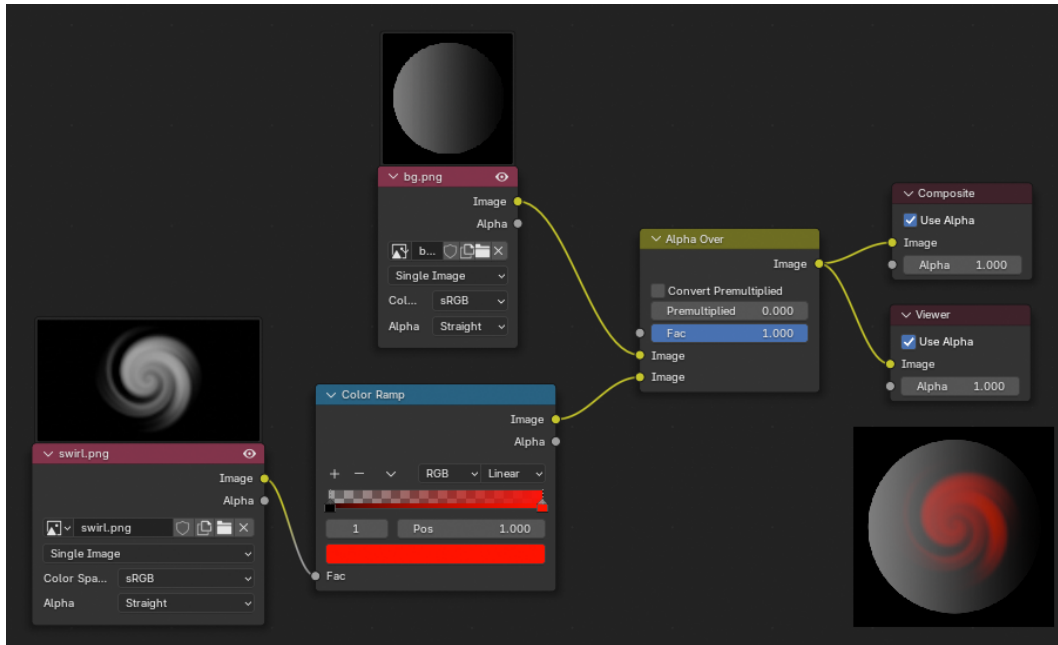
Image

The blended result.

Examples

Overlay

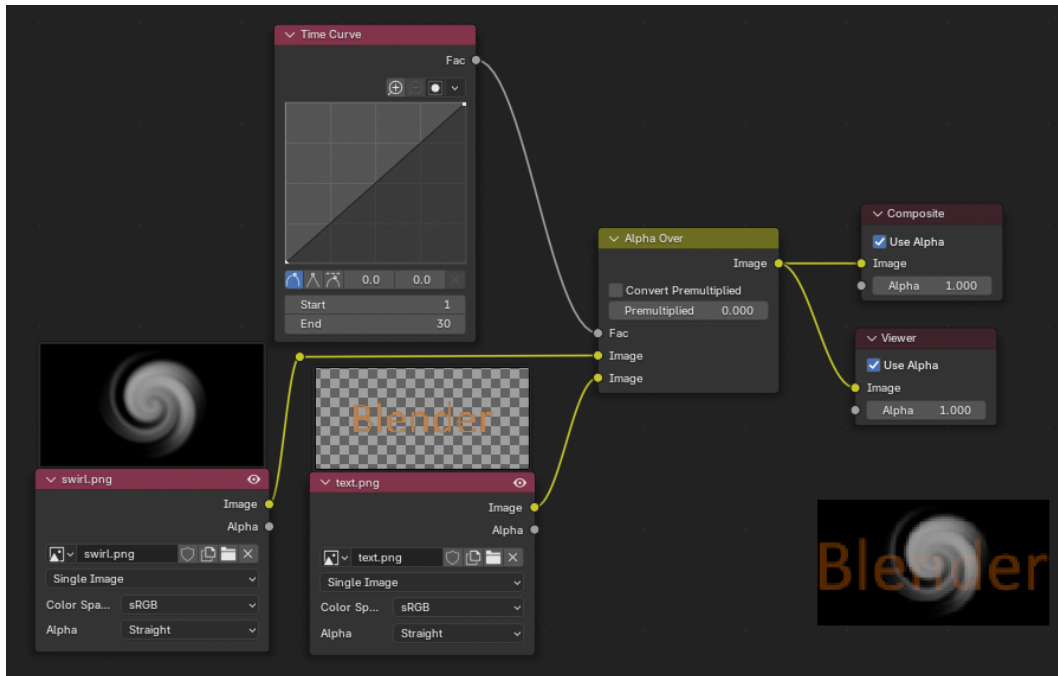
In the node tree below, the [Color Ramp Node](#) is used to convert an opaque, grayscale swirl image to a red one with transparency. Then, the *Alpha Over* node is used to overlay it on top of another image.



Assembling a composite image using Alpha Over.

Fade In

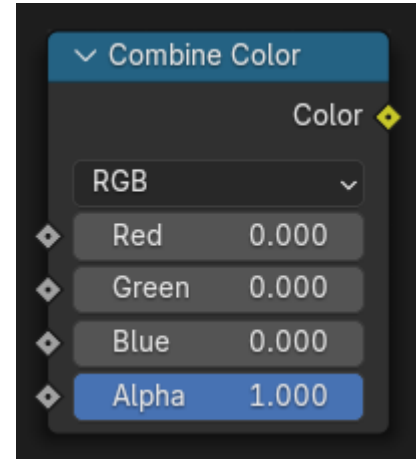
The example below uses the [Time Curve Node](#) to gradually increase the *Alpha Over* node's *Factor* from 0 to 1 over the course of 30 frames. This will result in the text fading in on top of the background image.



Animated fade in effect using Alpha Over.

Combine Color Node

The *Combine Color Node* combines an image from its composite color channels. The node can combine multiple [Color Models](#) depending on the Mode property.



Inputs

The outputs of this node depends on the Mode property (see below).

Alpha

The color channel that is responsible for the image's transparency.

Properties

Mode

The color model to output.

- | | |
|-----------------|--|
| RGB:: | Combine the three inputs: Red, Green, and Blue color channels into a single image. |
| HSV:: | Combine the three inputs: Hue, Saturation, and Value color channels into a single image. |
| HSL:: | Combine the three inputs: Hue, Saturation, and Lightness color channels into a single image. |
| YCbCrA:: | |

Combine the three inputs: Luminance, Chrominance Blue, and Chrominance Red color channels into a single image.

Color Space

ITU 601, ITU 709, JPEG

YUV::

Combine the three inputs: Luminance, U chrominance, and V chrominance color channels into a single image.

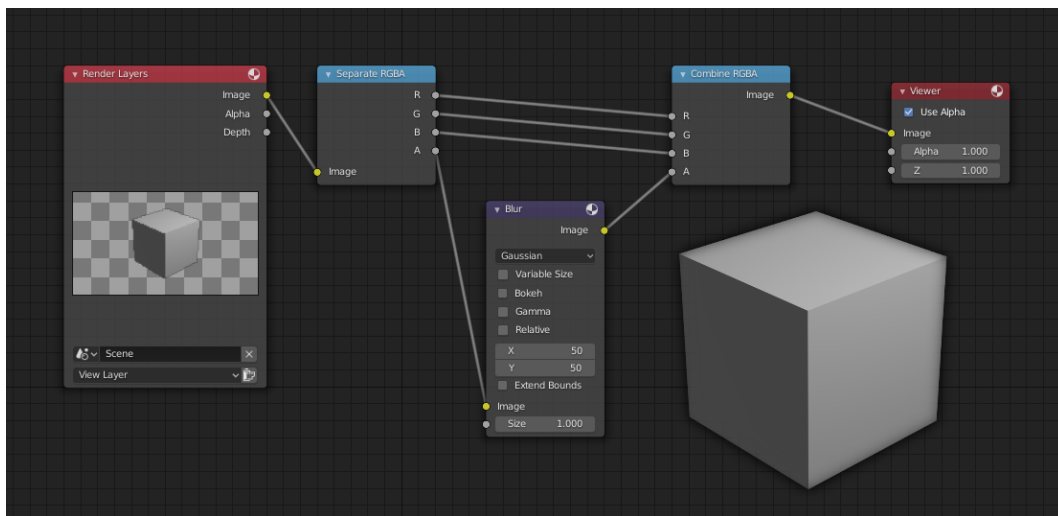
Output

Image

Standard image output.

Examples

Blur Alpha

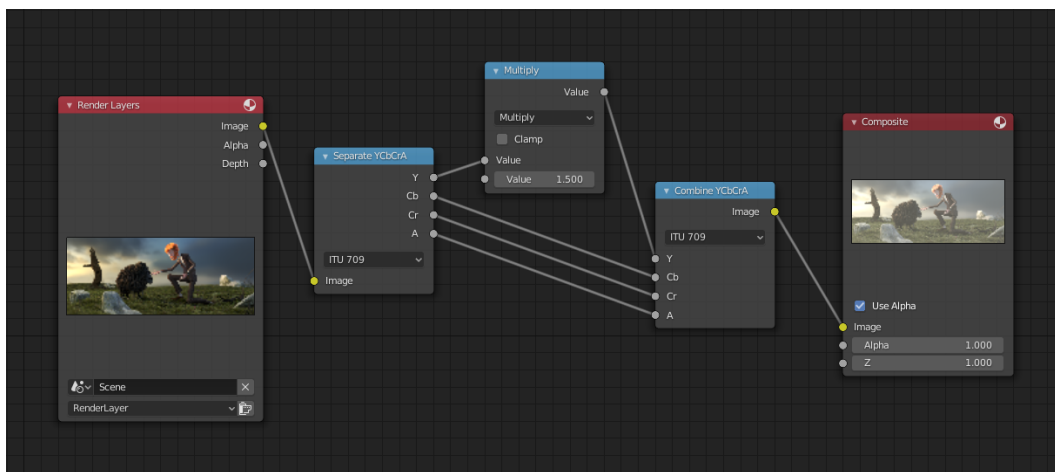


An example of blurring the alpha channel.

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like [Anti-Aliasing](#)

but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to “phase” in and out, as an “out-of-phase” time-traveling sync effect.

Increase Luminance



An example of the scaling the Luminance channel.

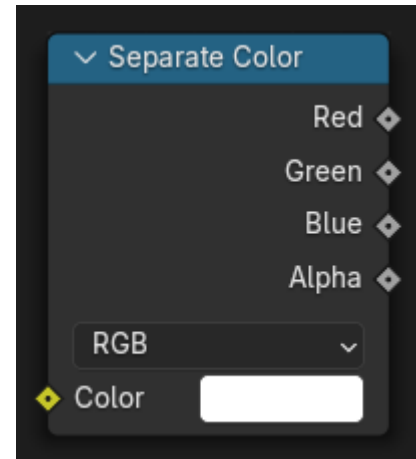
This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter.

Tip

If running these channels through a *Color Ramp* node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives a high-contrast effect.

Separate Color Node

The *Separate Color Node* splits an image into its composite color channels. The node can output multiple [Color Models](#) depending on the Mode property.



Inputs

Image
Standard image input.

Properties

Mode
The color model to output.

- | | |
|-----------------|---|
| RGB:: | Split the input image into it's three outputs: Red, Green, and Blue color channels. |
| HSV:: | Split the input image into it's three outputs: Hue, Saturation, and Value color channels. |
| HSL:: | Split the input image into it's three outputs: Hue, Saturation, and Lightness color channels. |
| YCbCrA:: | Split the input image into it's three outputs: Luminance, Chrominance Blue, and Chrominance Red color channels. |

Color Space
ITU 601, ITU 709, JPEG

YUV::

Split the input image into its three outputs:
Luminance, U chrominance, and V chrominance
color channels.

Outputs

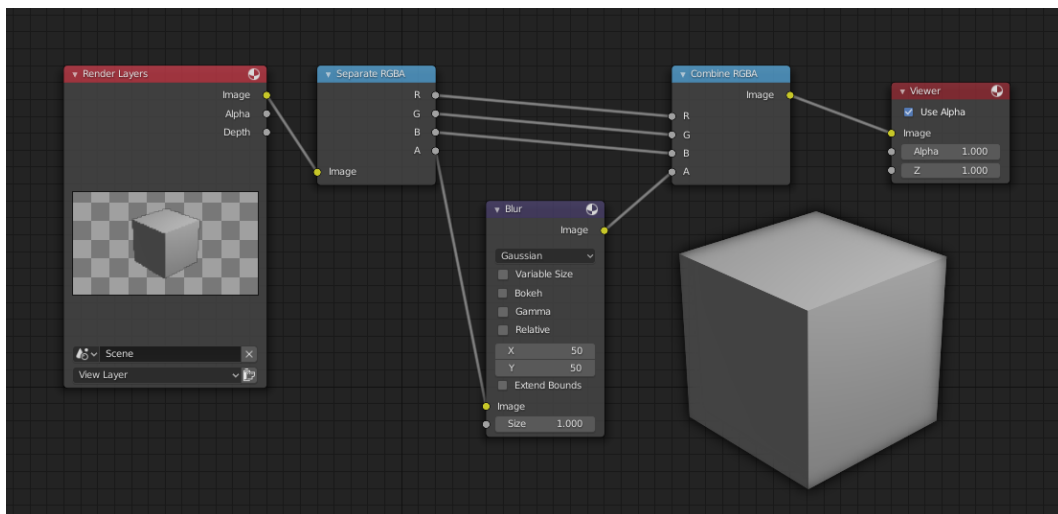
The outputs of this node depends on the Mode property (see above).

Alpha

The color channel that is responsible for the image's transparency.

Examples

Blur Alpha

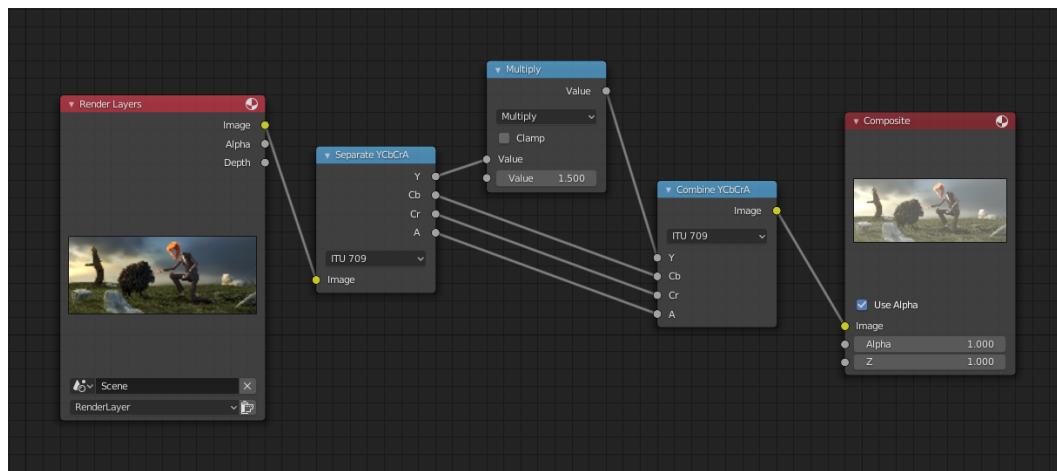


An example of blurring the alpha channel.

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like [Anti-Aliasing](#) but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a

broader scale will make the object appear to “phase” in and out, as an “out-of-phase” time-traveling sync effect.

Increase Luminance



An example of the scaling the Luminance channel.

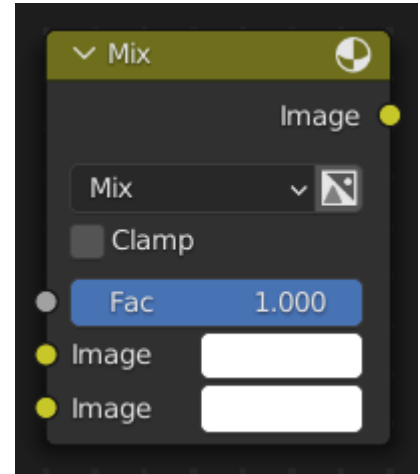
This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter.

Tip

If running these channels through a *Color Ramp* node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives a high-contrast effect.

Mix Node

Blends two images together, much like how an image editing program blends two layers.



Inputs

Fac

The opacity of the foreground image.

Image

The background image. Determines the dimensions of the output.

Image

The foreground image.

Keep in mind that, unlike image editing programs where the foreground layer is on top, the foreground slot in Blender is on the bottom.

Properties

Blending Mode

The [blending mode](#) to use.

Mix

Regular alpha blending. Typically called *Normal* in image editing programs.

Darken

For each color component, takes the smallest of the two values being blended.

Multiply

Multiplies the colors component by component. Blending with a white pixel (value 1.0) has no effect, while blending with a black one (0.0) always results in black.

Color Burn

Inverts the background color, divides it by the foreground color, and inverts the result.

Lighten

For each color component, takes the largest of the two values being blended.

Screen

Inverts both colors, multiplies them, and inverts the result.

Color Dodge

Divides the background color by the inverted foreground color.

Add

Adds the two colors together.

Overlay

Applies Multiply blending if the foreground color's lightness is below 0.5, or Screen blending if it's above.

Soft Light

Like Overlay, but more subtle.

Linear Light

Applies Linear Burn blending ($\text{background} + \text{foreground} - 1$) if the foreground color's lightness is below 0.5, or Linear Dodge ($\text{background} + \text{foreground}$) if it's above.

Difference

For each component, subtracts the lower value from the higher value.

Exclusion

Adds the two colors, then subtracts their multiple twice.

Subtract

Subtracts the foreground color from the background color.

Divide

Divides the background color by the foreground color.

Hue

Combines the saturation and value of the background color with the hue of the foreground color.

Saturation

Combines the hue and value of the background color with the saturation of the foreground color.

Color

Combines the value of the background color with the hue and saturation of the foreground color.

Value

Combines the hue and saturation of the background color with the value of the foreground color.

Use Alpha

Whether to use the alpha channel of the foreground image during mixing. The alpha channel of the background image is always used.

Clamp

Clamp the output value to the $[0.0, 1.0]$ range.

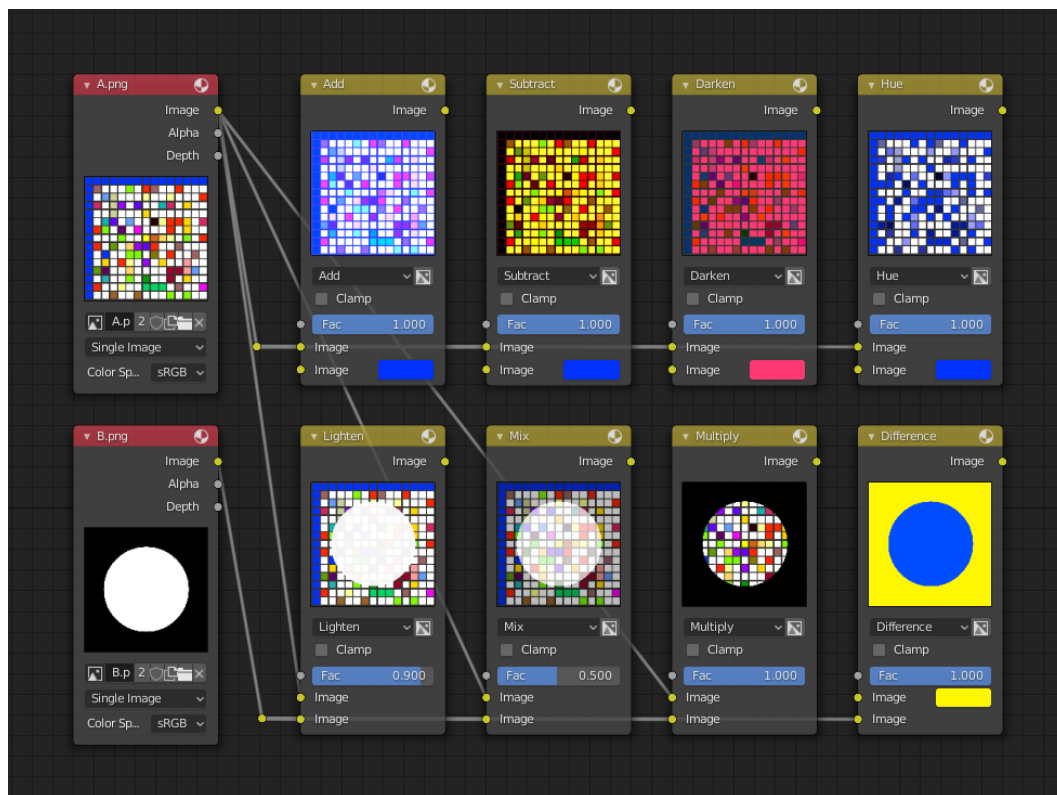
Outputs

Image

The result of the mixing operation.

Examples

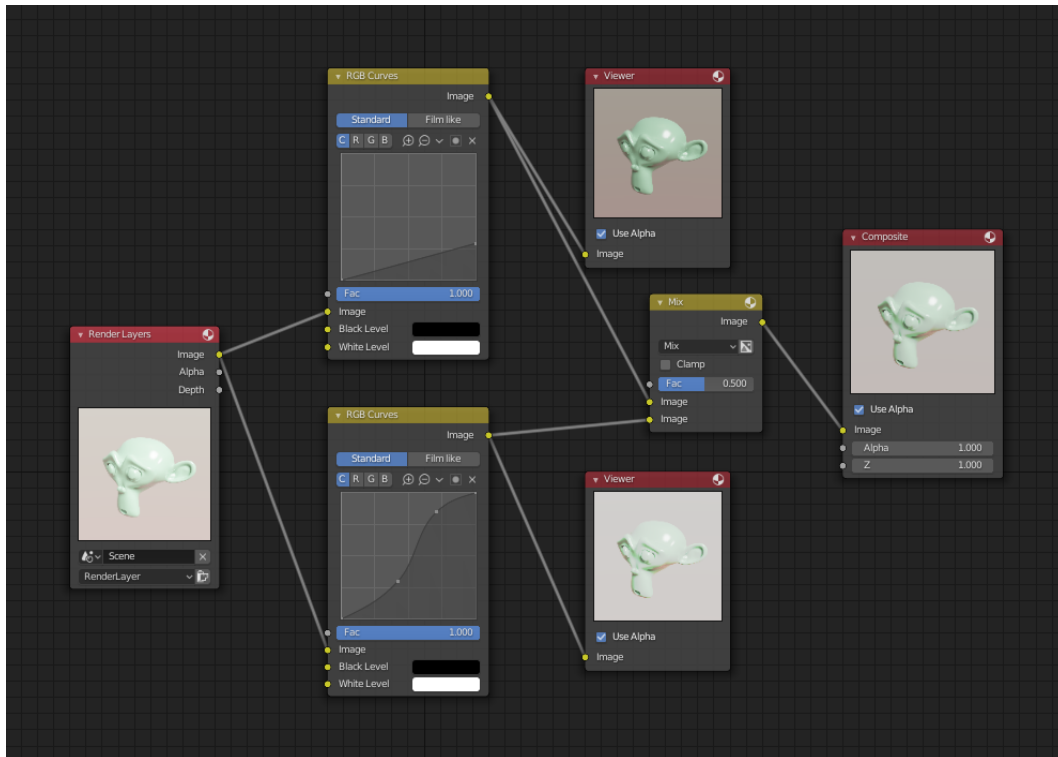
Below are examples of blending modes, as well as some practical use cases.



Blending a colored pattern with a flat color (top row) and a circular mask (bottom row).

Fixing overexposure

The Compositing setup below shows how to fix an overexposed render by darkening it and increasing contrast.



Example node setup showing two RGB Curves nodes and a Mix node for composition.

The top [RGB Curves Node](#) darkens the image by linearly scaling each color value to a smaller one.

The bottom curve node increases contrast by making small values smaller and large values larger.

Finally, the Mix node blends the two together.

Watermark Images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising.

Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting

service, or simply to track their image's proliferation throughout the web.

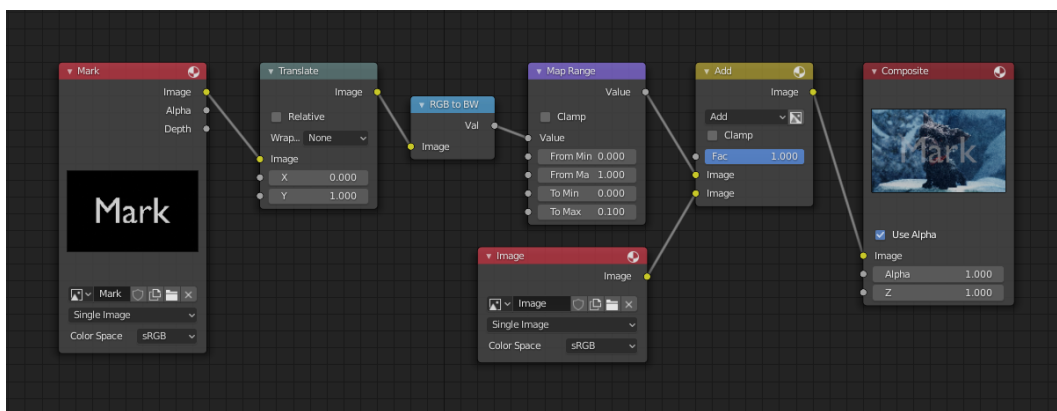
Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you.

In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the *RGB to BW* node to convert the color image to grayscale numbers, which we then feed into the *Map Range* node to reduce the mark to one-tenth of its original intensity.

The *Add* node (*Mix* node with blending mode *Add*) adds the corresponding pixels, making the ones containing the mark ever-so-slightly brighter.

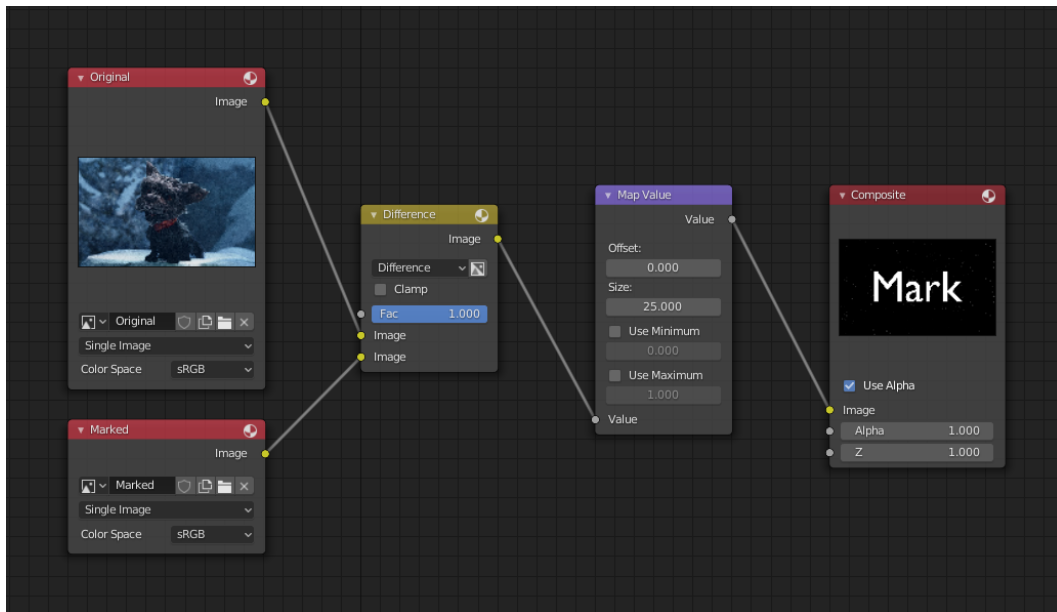


Embedding a watermark in an image.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Decoding an Image for your Watermark

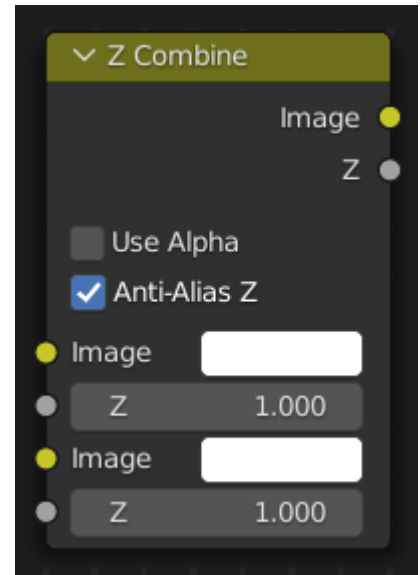
When you see an image that you think might be yours, use the node tree below to compare it to your stock image (pre-watermarked original). In this tree, the *Mix* node is set to Difference, and the *Map Value* node amplifies any difference. You can see how the original mark clearly stands out.



Checking an image for your watermark.

Z Combine Node

The Z Combine node combines two images based on their Z-depth maps. It overlays the images using the provided Z values to detect which parts of one image are in front of the other.



Inputs

Image

The background image.

Z

Z depth of the background image.

Image

The foreground image.

Z

Z depth of the foreground image.

Properties

Use Alpha

The chosen Image pixel alpha channel is also carried over. If a pixel is partially or totally transparent, the result of the Z Combine will also be partially transparent; in which case the background image will show through the foreground (chosen) pixel.

Anti-Alias Z

Applies [Anti-Aliasing](#) to avoid artifacts at sharp edges or areas with a high contrast.

Outputs

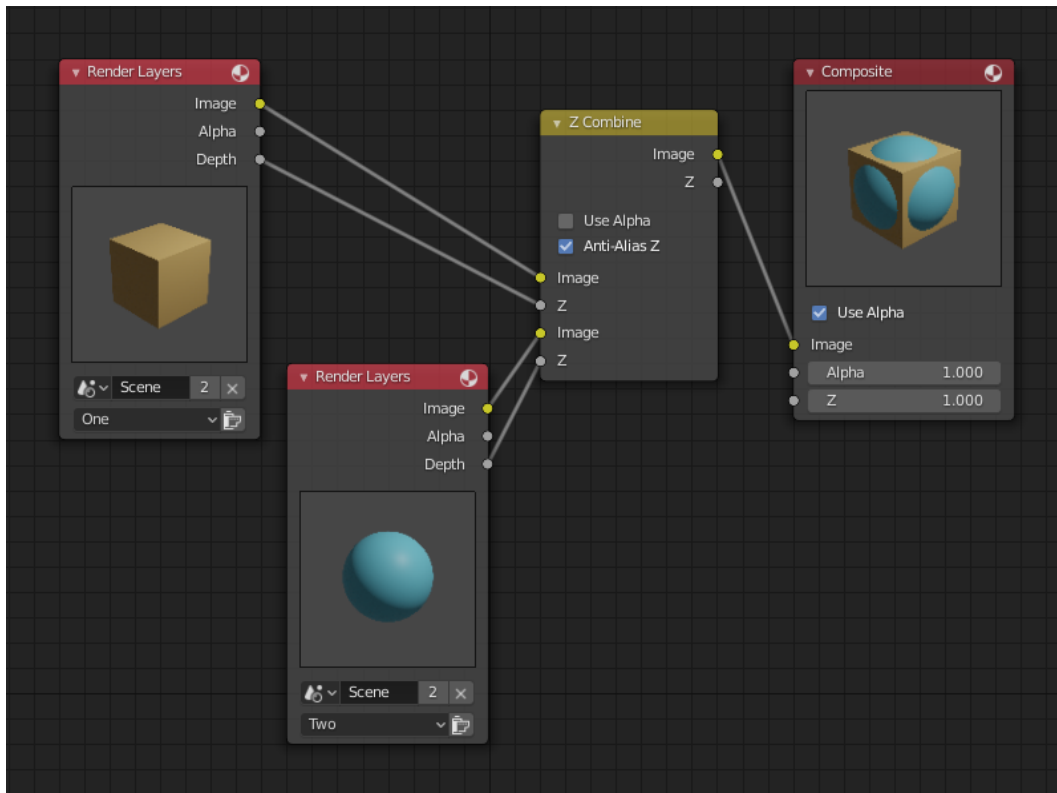
Image

If both Z values are equal, it will use the foreground image. Whichever Z value is less decides which image pixel is used. See [Z-buffer](#).

Z

The combined Z depth, which allows to thread multiple Z-combines together.

Examples



Choosing closest pixels.

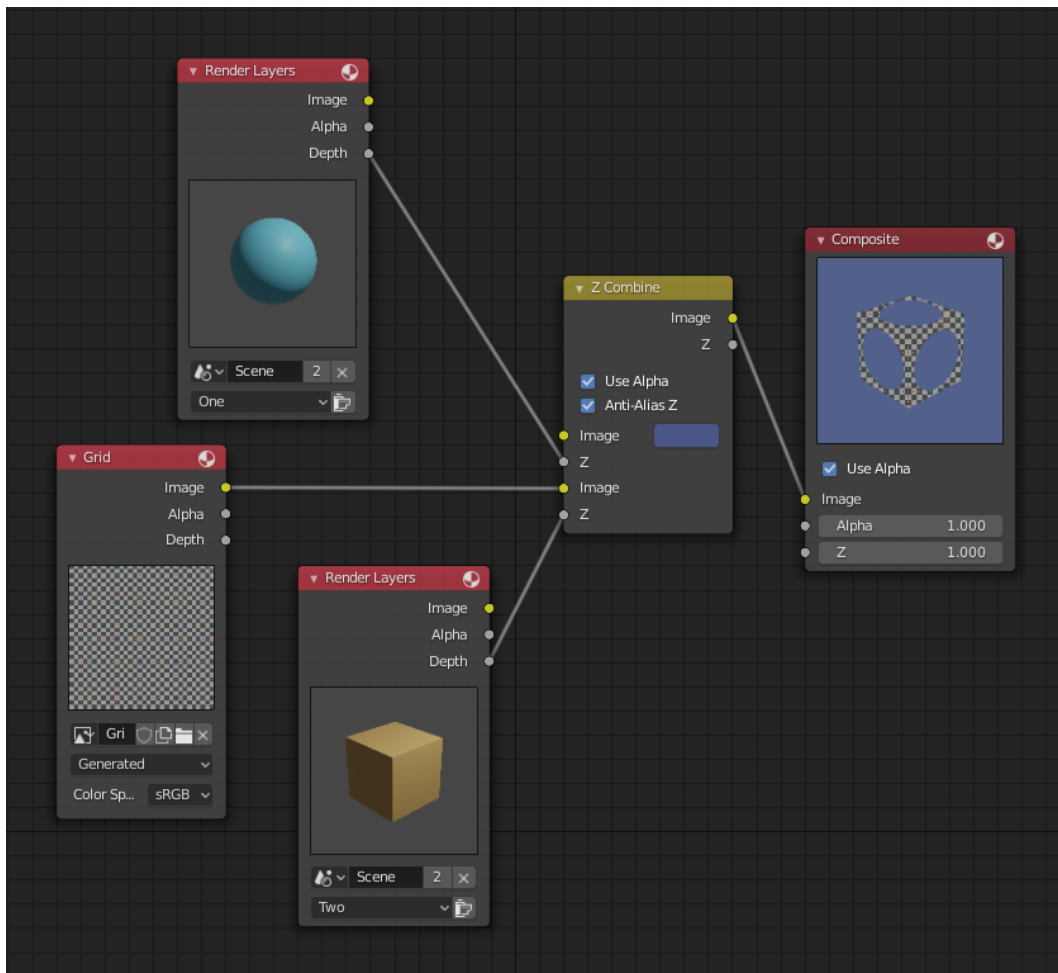
In the example above, the render output from two scenes are mixed using the Z Combine node, one from a sphere of size 1.3, and the other a cube of size 1.0. The sphere and square are located at the same place. The cube is tipped forward, so the corner in the center is closer to the camera than the sphere surface; so Z Combine chooses to use the cube's pixels. But the sphere is slightly larger (a size of 1.3 versus 1.0), so it does not fit totally inside the cube. At some point, as the cube's sides recede back away from the camera, the sphere's sides are closer. When this happens, Z Combine uses the sphere's pixels to form the resulting picture.

This node can be used to combine a foreground with a background matte painting. Walt Disney pioneered the use of multi-plane mattes, where three or four partial mattes were painted on glass and placed on the left and right at different Z positions; minimal camera moves to the right created the illusion of depth as Bambi moved through the forest.

Note

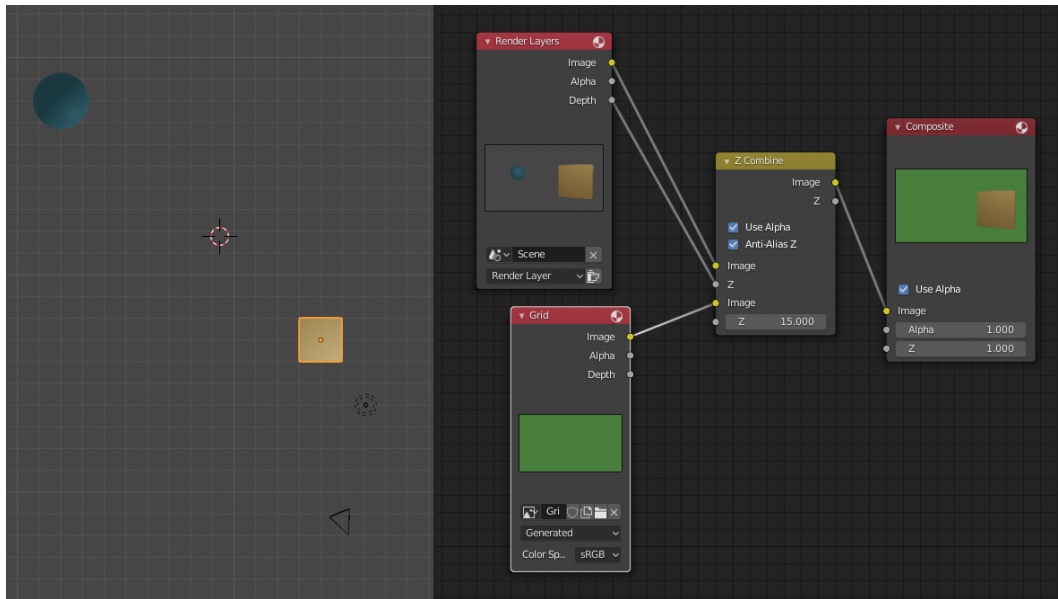
Valid Input

Z Input Sockets do not accept fixed values; they must get a vector set (see Map Value node). Image Input Sockets will not accept a color since they do not have UV coordinates.



Mix and match images.

The Z Combine can be used to merge two images as well. Using the Z values from the sphere and cube scenes above, but inputting different images, yields the example to the right.



Z Combine in action.

In this node setup a render scene is mixed with a flat image. In the side view of the scene, the orange cube is 10 units away from the camera, and the blue ball is 20. The 3D cursor is about 15 units away from the camera. The image is Z-in at a location of 15, thus inserting it in between the cube and the ball. The resulting image appears to have the cube on the green image.

Note

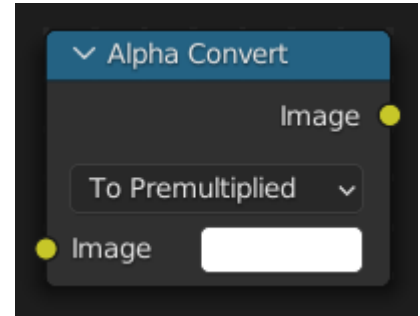
Invisible Man Effect

If a foreground image with a higher Alpha than the background, is then mixed in the Z Combine with a slightly magnified background, the outline of the transparent area will distort the background, enough to make it look like seeing a part of the background through an invisible yet Fresnel-lens object.

Alpha Convert Node

The *Alpha Convert Node* converts the alpha channel format of an image.

For compositing and rendering, [Premultiplied Alpha](#) is the standard in Blender. Render layers will be premultiplied alpha, and images loaded into rendering or compositing will be converted to this.



If you want to do a compositing operation with straight alpha, the *Alpha Convert* node can be used. Typically this would be a color correction operation where it might give better results working on RGB channels without alpha. If the alpha is converted to straight in the Compositor, it should be converted back to premultiplied before the *Composite Output* node, otherwise some artifacts might occur.

Inputs

Image

Standard color input.

Properties

Mapping

The direction of convert alpha. For details on the difference between both ways to store alpha values see [Alpha Channel](#).

To Premultiplied:: Converts from [Straight Alpha](#) to [Premultiplied Alpha](#).

To Straight:: Converts from [Premultiplied Alpha](#) to [Straight Alpha](#).

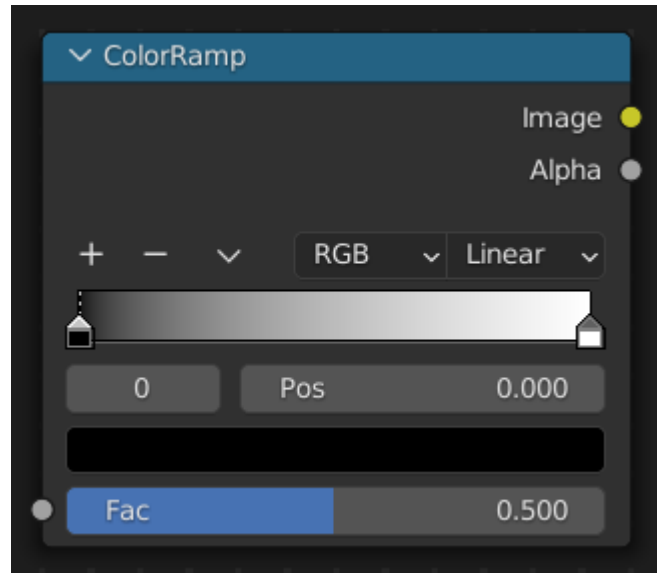
Outputs

Image

Standard color output.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Inputs

Factor

The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp

See [Color Ramp Widget](#).

Outputs

Image/Color

Standard color output.

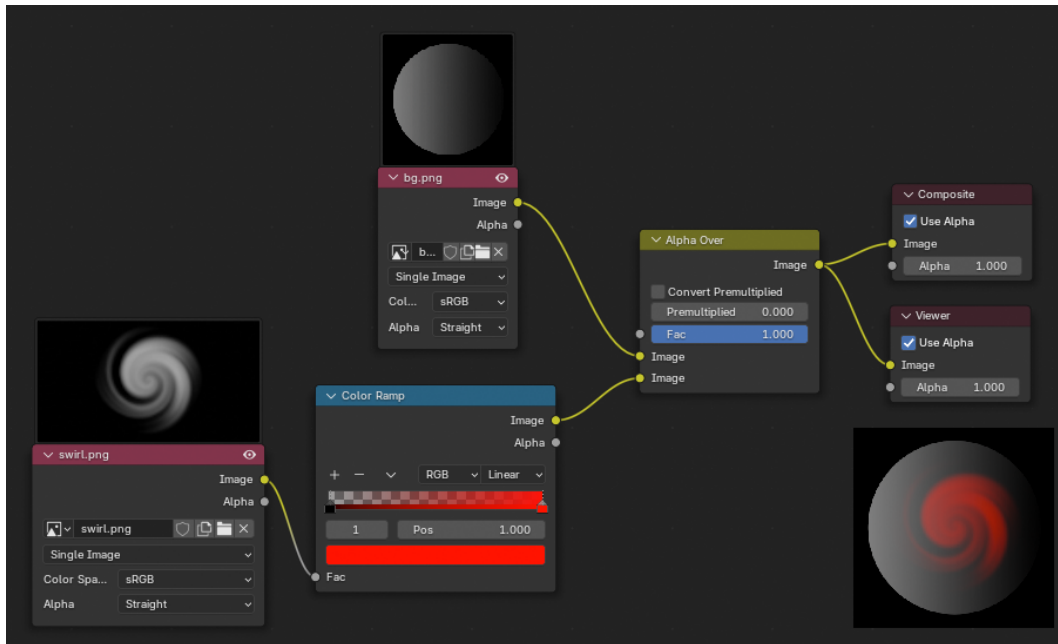
Alpha

Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

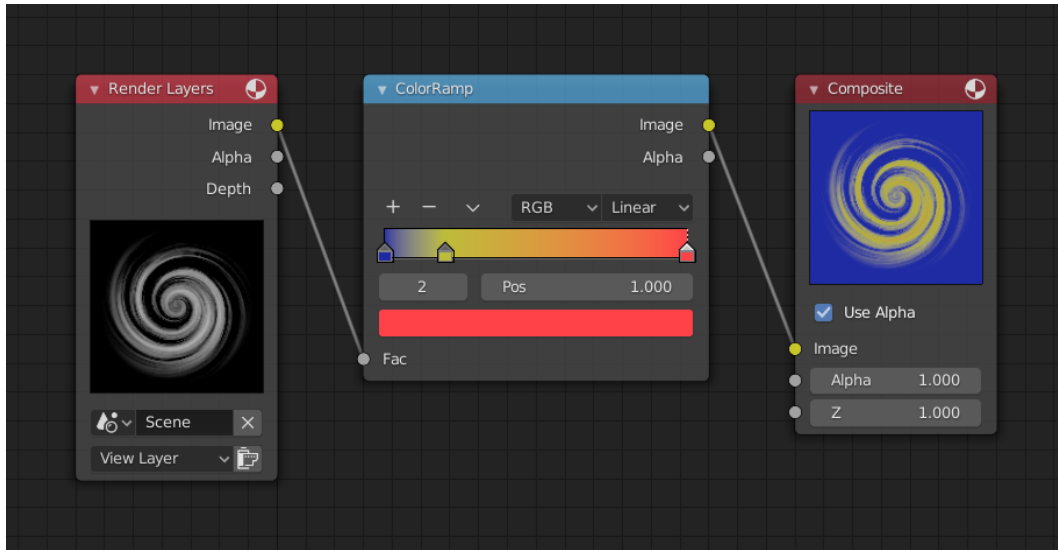


In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.



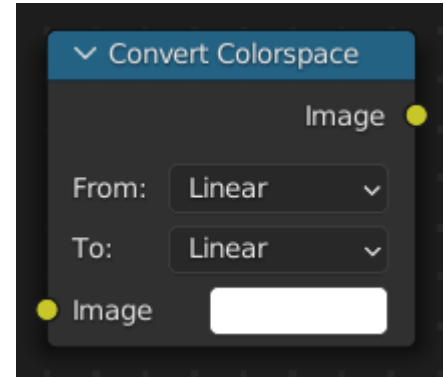
The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Convert Colorspace Node

The *Convert Colorspace* node converts images between [color spaces](#).

Note

Images are already automatically converted into linear color space unless specified in the image's [Color Space](#) option.



Inputs

Image

Standard color input.

Properties

From, To

The color space of the input image and the color space to convert it to.

The list of color spaces depends on the active [OCIO config](#). The default supported color spaces are described in detail here: [Default OpenColorIO Configuration](#)

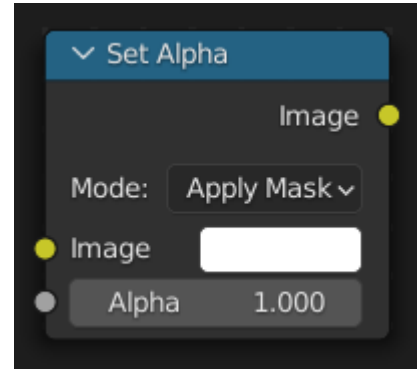
Outputs

Image

Standard color output.

Set Alpha Node

The *Set Alpha Node* adds an alpha channel to an image.



Inputs

Image

Standard color input.

Alpha

The amount of Alpha can be set for the whole image by using the input field or per pixel by connecting to the socket.

Properties

Mode

Method to mix the alpha with the input *Image*.

Apply Mask::

Multiply the input images RGBA channels by the *Alpha* input value. In this cases, the output is using [Premultiplied Alpha](#).

Replace Alpha::

Replace the inputs alpha channel with the *Alpha* input value. In this cases, the output is using [Straight Alpha](#).

Outputs

Image

Standard color output.

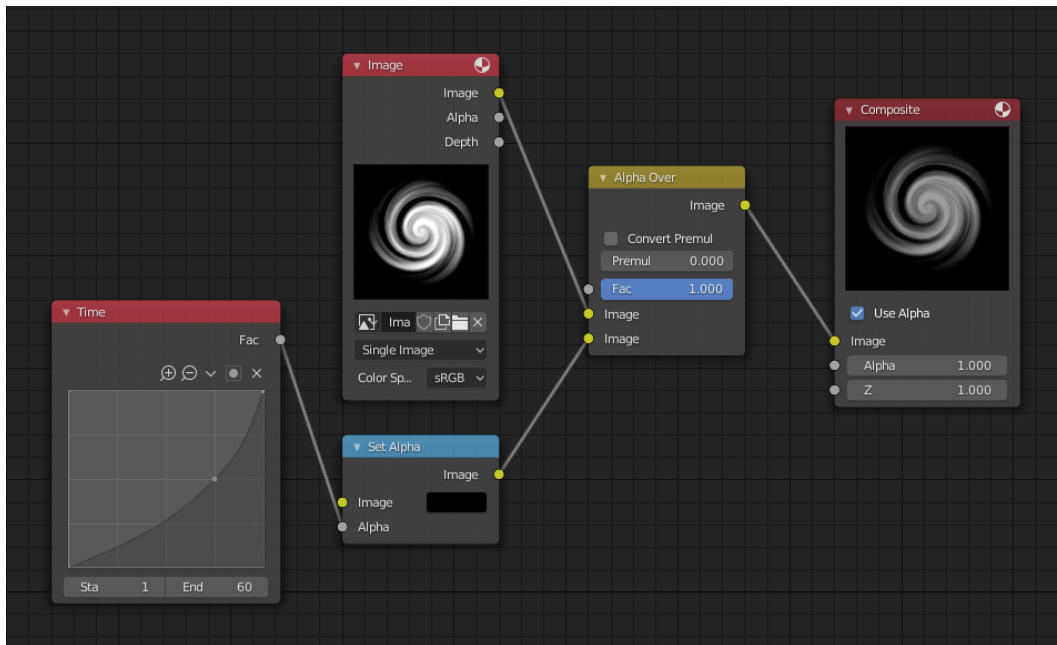
Note

This is not, and is not intended to be, a general-purpose solution to the problem of compositing an image that does not contain alpha information. You might wish to use “chroma keying” or “difference keying” (as discussed elsewhere) if you can. This node is most often used (with a suitable input being provided by means of the socket) in those troublesome cases when you *cannot*, for some reason, use those techniques directly.

Example

Fade To Black

To transition the audience from one scene or shot to another, a common technique is to “fade to black”. As its name implies, the scene fades to a black screen. You can also “fade to white” or whatever color you wish, but black is a good neutral color that is easy on the eyes and intellectually “resets” the viewer’s mind. The node tree below shows how to do this using the Set Alpha node.



Fade to black.

In the example above, the alpha channel of the swirl image is ignored. Instead, a [Time Curve node](#) introduces a factor from 0.0 to 1.0 over 60 frames, or about 2 seconds, to the Set Alpha node. Note that the time curve is exponentially-shaped, so that the overall blackness will fade in slowly and then accelerate toward the end. The Set Alpha node does not need an input image; instead, the flat (shadeless) black color is used. The Set Alpha Node uses the input factor and color to create a black image that has an alpha set which goes from 0.0 to 1.0 over 60 frames, or completely transparent to completely opaque. Think of alpha as a multiplier for how vivid you can see that pixel. These two images are combined by the Alpha Over node completely (a *Factor* of 1.0) to produce the composite image. The Set Alpha node will thus, depending on the frame being rendered, produce a black image that has some amount of transparency. Setup and animate, and you have an image sequence that fades to black over a two-second period.

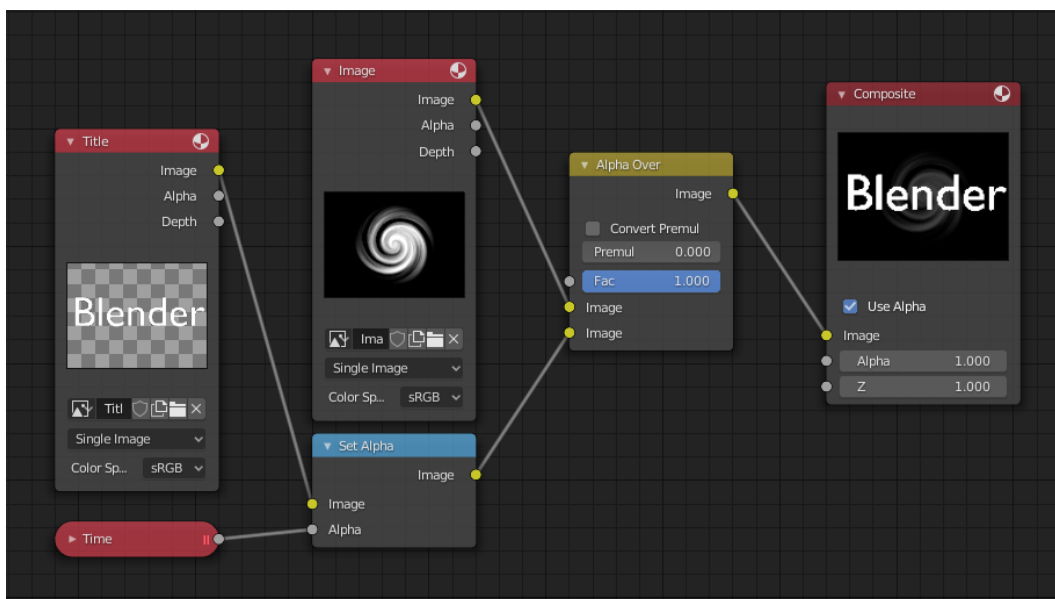
Note

No Scene Information Used

This example node tree does not use the Render Layer node. To produce this 2-second animation, no Blender scene information was used. This is an example of using Blender’s powerful compositing abilities separate from its modeling and animation capabilities. (A Render Layer could be substituted for the Image layer, and the “fade-network” effect will still produce the same effect.)

Fade In a Title

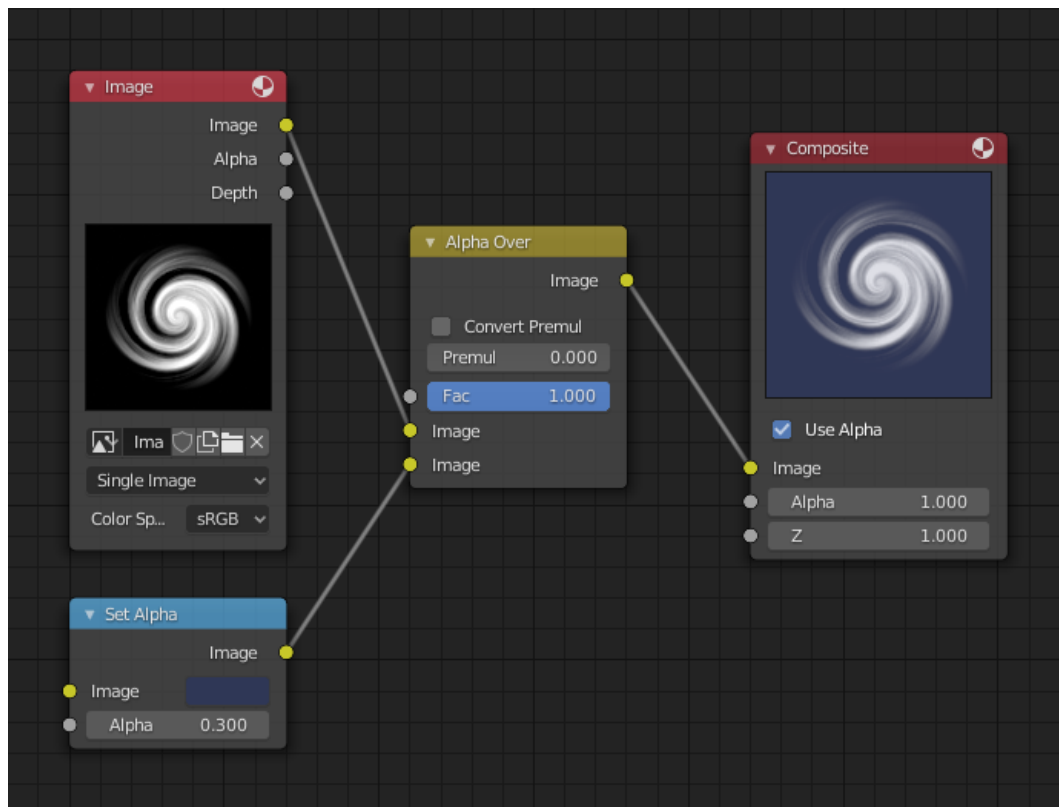
To introduce your animation, you will want to present the title of your animation over a background. You can have the title fly in, or fade it in. To fade it in, use the Set Alpha node with the Time Curve node as shown below.



Using Set Alpha to fade in a title.

In the above example, a Time curve provides the Alpha value to the input socket. The current Render Layer node, which has the title in view, provides the image. As before, the Alpha Over node mixes (using the alpha values) the background swirl and the alpha title to produce the composite image.

Colorizing a BW Image



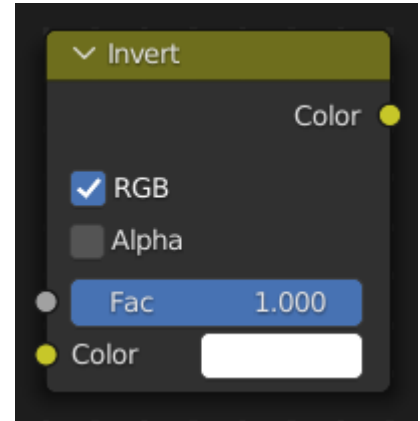
Using Set Alpha to colorize an image.

In the example above, notice how the blue tinge of the render input colors the swirl. You can use the Set Alpha node's color field with this kind of node tree to add a consistent color to a BW image.

In the example tree to the right, use the *Alpha* value of the Set Alpha node to give a desired degree of colorization. Thread the input image and the Set Alpha node into an Alpha Over node to colorize any black-and-white image in this manner.

Invert Color Node

Inverts the colors in the input image, producing a negative.



Inputs

Factor

The amount of influence the node exerts on the image.

Color

Standard color input.

Properties

In the compositing context, this node has the following properties:

RGB

Invert the color channels.

Alpha

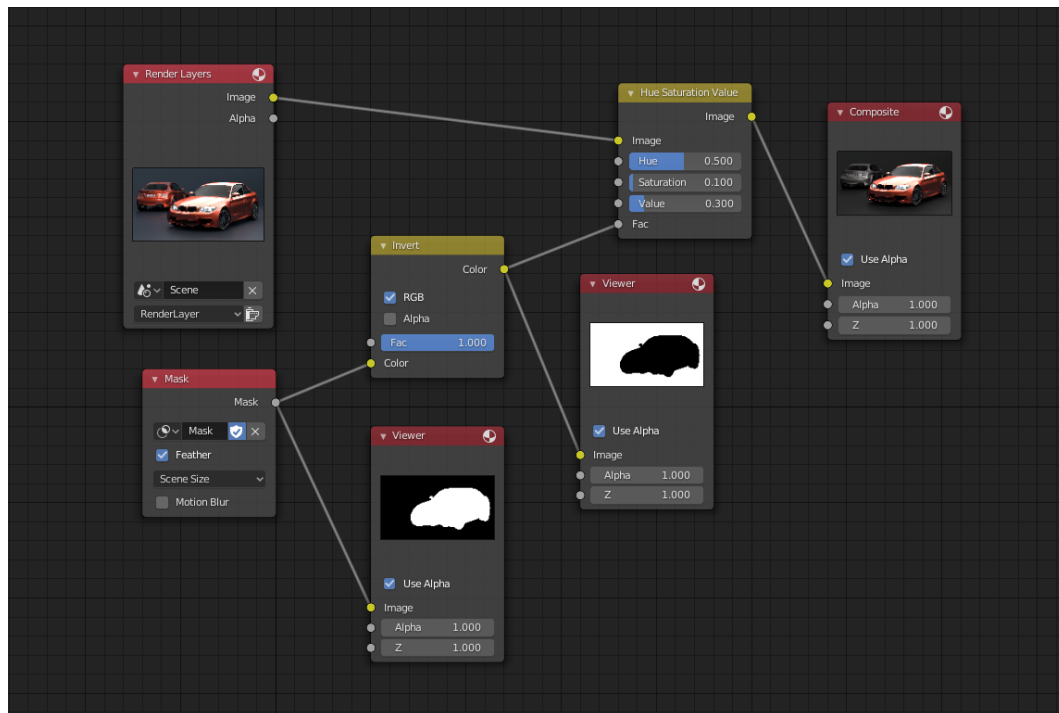
Invert the alpha channel.

Outputs

Color

Standard color output.

Example



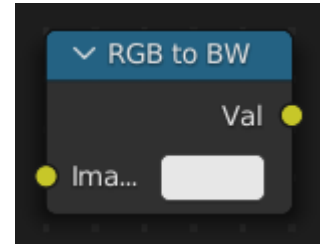
The Invert node is used to invert the mask.

RGB to BW Node

The *RGB to BW Node* makes a color image black-and-white by outputting its luminance.

Note

You can directly connect Color sockets to Value sockets in node graphs, which also converts the image to black-and-white. As such, this node is not always necessary.



Inputs

Image

Color image input.

Properties

This node has no properties.

Outputs

Value

Grayscale value output.

Filter Nodes

Filters process the pixels of an image to highlight additional details or perform some sort of post-processing effect on the image.

- [Blur Filter Nodes](#)
-

- [Anti-Aliasing Node](#)
 - [Denoise Node](#)
 - [Despeckle Node](#)
-

- [Dilate/Erode Node](#)
 - [Inpaint Node](#)
-

- [Filter Node](#)
- [Glare Node](#)
- [Kuwahara Node](#)
- [Pixelate Node](#)
- [Posterize](#)
- [Sun Beams Node](#)

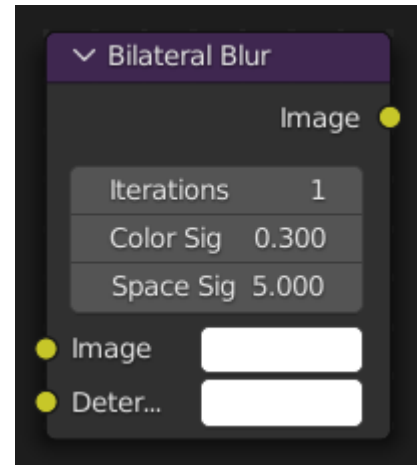
Blur Filter Nodes

- [Bilateral Blur Node](#)
- [Blur Node](#)
- [Bokeh Blur Node](#)
- [Defocus Node](#)
- [Directional Blur Node](#)
- [Vector Blur Node](#)

Bilateral Blur Node

The Bilateral Blur node performs a high-quality adaptive blur, blurring the image while retaining sharp edges.

It can be used for various purposes like: smoothing noisy render passes to avoid longer computation times in example ray-traced ambient occlusion, blurry refractions/reflections, soft shadows, or to make non-photorealistic compositing effects.



Inputs

Image

Standard color input. If only the image input is connected, the node blurs the image depending on the edges present in the source image.

Determinator

Which is non-obligatory and if the Determinator is connected, it serves as the source for defining edges/borders for the blur in the image. This has great advantage in case the source image is too noisy, but normals in combination with Z-buffer can still define exact borders/edges of objects.

Properties

Iterations

Defines how many times the filter should perform the operation on the image. It practically defines the radius of blur.

Color Sigma

Defines the threshold for which color differences in the image should be taken as edges.

Space Sigma

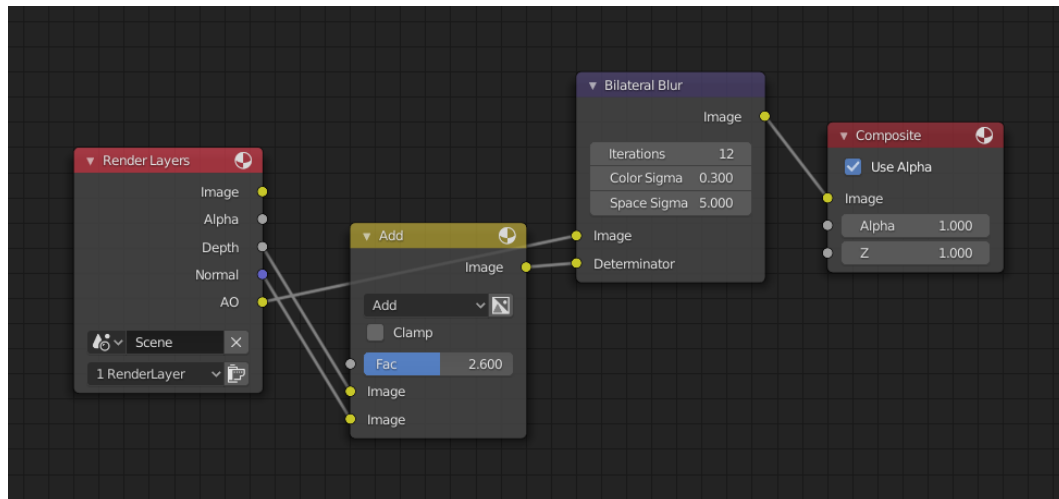
A fine-tuning variable for blur radius.

Outputs

Image

Standard color output.

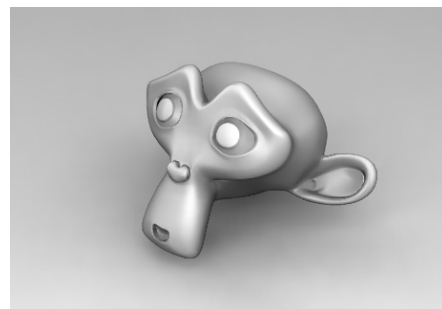
Example



Bilateral smoothed Ambient Occlusion. [blend-file example](#)



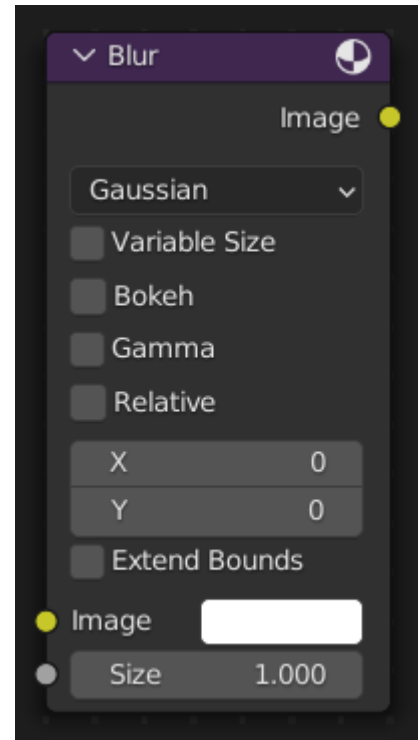
Render result.



Composite.

Blur Node

The Blur node blurs an image, providing several blur modes.



Inputs

Image

Standard color input.

Size

The optional Size input will be multiplied with the X and Y blur radius values. It also accepts a value image, to control the blur radius with a mask. The values should be mapped between (0 to 1) for an optimal effect.

Properties

Type

The difference between the types is in the way they handle sharp edges, smooth gradients and preserve the highs and the lows.

Flat::	Simply blurs everything uniformly.
Tent::	Preserves the high and the lows better by making a linear falloff.
Quadratic::	Looks similar to <i>Gaussian</i> but can be a little faster but slightly worse looking.
Cubic::	Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.
Gaussian::	Gives the best looking results but tends to be the slowest.
Fast Gaussian::	An approximation of the Gaussian.
Catmull-Rom::	Catmull-Rom keeps sharp contrast edges crisp.
Mitch::	Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.

Variable Size

Allows a variable blur radius, if the size input is an image.

Bokeh

The Bokeh button will force the Blur node to use a circular blur filter. This gives higher quality results, but is slower than using a normal filter.

Gamma

The Gamma button applies a gamma correction on the image before blurring it.

Relative

Percentage Value of the blur radius relative to the image size.

Aspect Correction

None, Y, X

X, Y

Values set the ellipsoid radius in numbers of pixels over which to spread the blur effect.

Extend Bounds

Allows the image, that is being blurred, to extend past its original dimension.

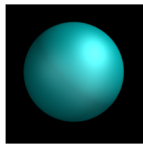
Outputs

Image

Standard color output.

Example

Blur node blur modes using 20% of image size as XY, no Bokeh/Gamma.



Original
image.



Flat.



Tent.



Quadratic.



Cubic.



Gaussian.



Fast
Gaussian.



Catmull-
Rom.

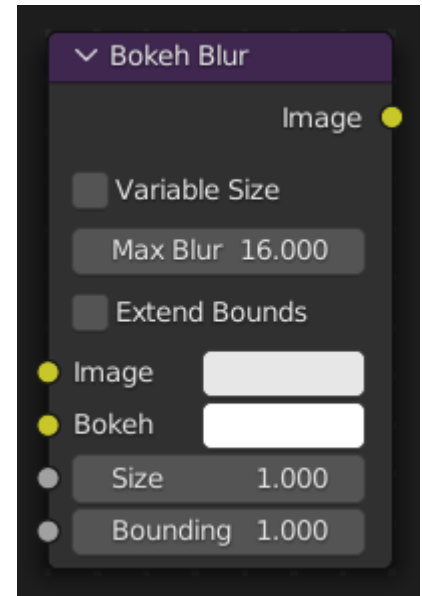


Mitch.

Bokeh Blur Node

The Bokeh Blur node generates a bokeh type blur similar to Defocus. Unlike defocus an in-focus region is defined in the Compositor. There is also more flexibility in the type of blur applied through the [Bokeh Image](#) node.

Several performance optimizations are also available such calculation area restriction and masking.



Inputs

Image

Standard color input.

Bokeh

This is an input for the [Bokeh Image](#) node.

Size

Size controls the amount of blur. Size can either be a single value across the entire image or a variable value controlled by an input image. In order to use the latter, the Variable Size option must be selected. See the examples section below for more on how to use this.

Bounding Box

This can be used with a [Box Mask](#) matte node or with a [Mask](#) input node to restrict the area of the image the blur is applied to. This could be helpful, for example, when developing a node system by allowing only

a small area of the image to be filtered thus saving composite time each time adjustments are made.

Properties

Variable Size

Allows a variable blur radius, if the Size input is an image.

Max Blur

Max Blur is intended to act as an optimization tool by limiting the number of pixels across which the blur is calculated.

Outputs

Image

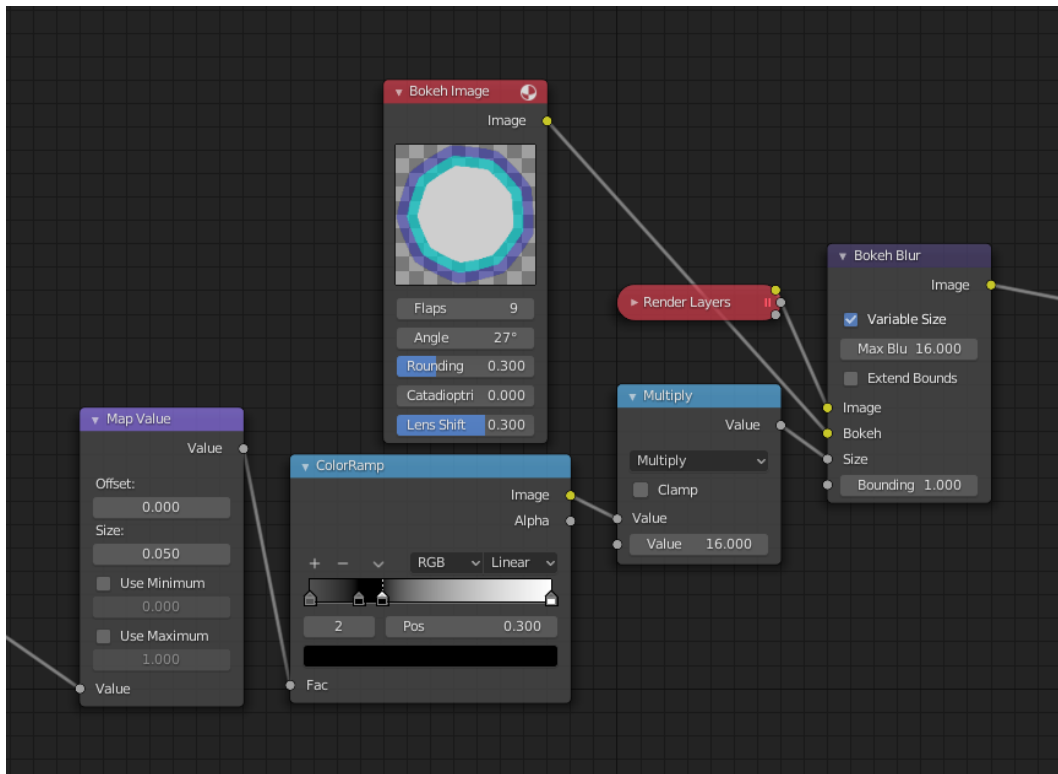
Standard color output.

Examples

Three examples of how the size input may be used follow.

An [ID masked](#) alpha image can be used so that a background is blurred while foreground objects remain in focus. To prevent strange edges the [Dilate Node](#) should be used.

The Z pass can be visualized using a [Map Value](#) node and a [Color Ramp](#) node as described in [Render Layers](#). A *multiply* [Math](#) node can be used following the color ramp so that a blur value greater than one is used for objects outside the focal range.



Z pass used.

A manually created grayscale image can be used to define the sharp and blurry areas of a preexisting image. Again, a Multiply Node can be used so that a blur value greater than one is used.

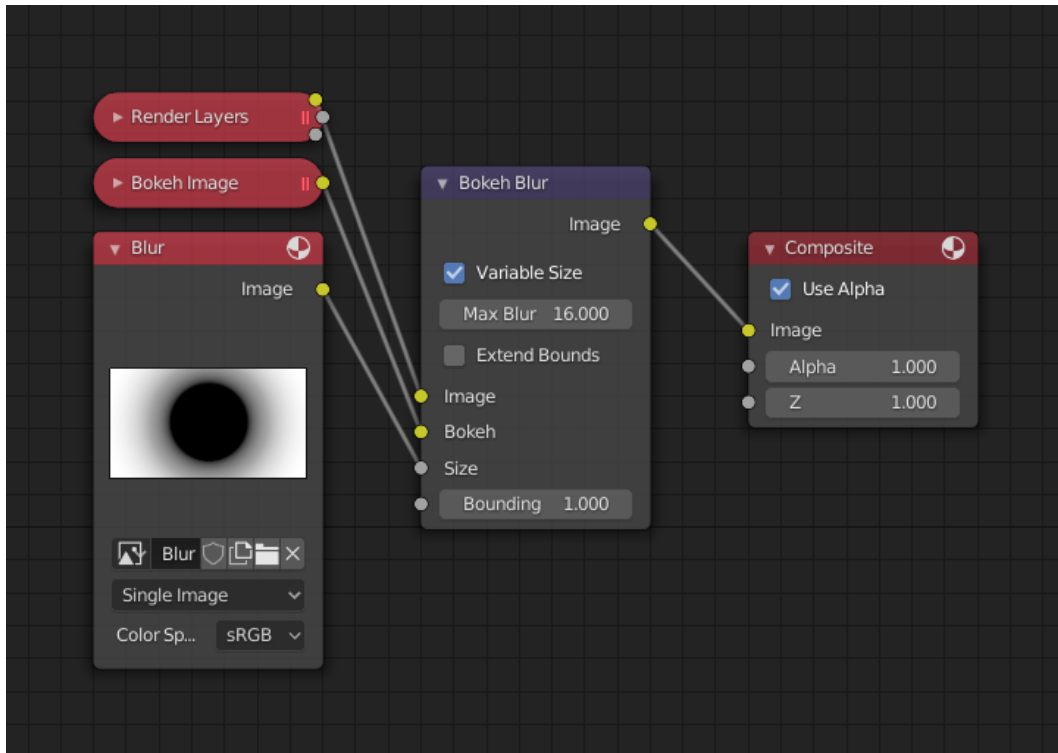


Image used.



Z pass used.

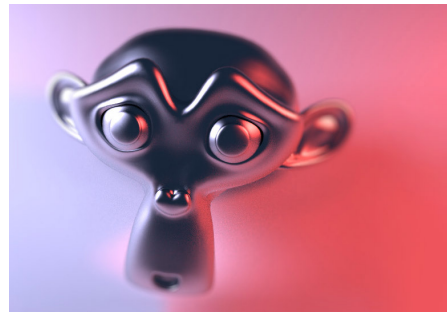
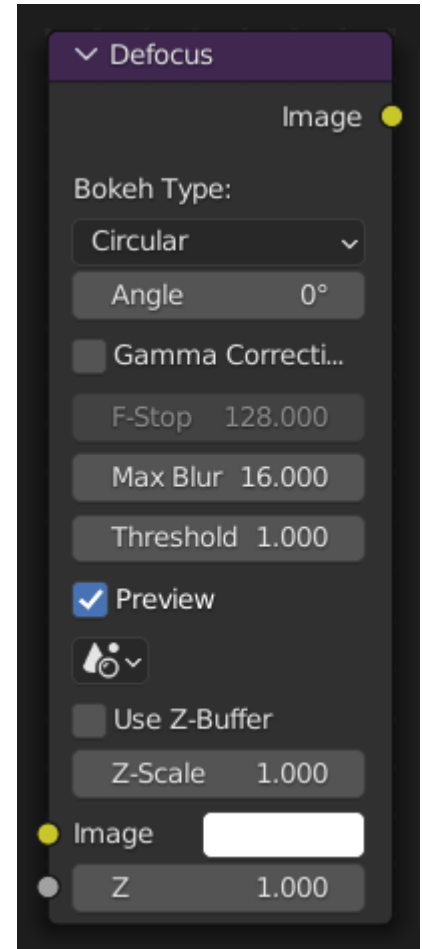


Image used.

Defocus Node

The *Defocus Node* blurs areas of an image based on a Z depth map or mask input.

It is typically used to emulate depth of field ([DOF](#)) using a post-processing method with a Z-buffer input. But also allows to blur images that are not based on Z depth too.



Inputs

Image

Standard color input.

Z

Z-buffer input, but could also be a (grayscale) image used as a mask, or a single value input.

Properties

Bokeh Type

The number of iris blades of the virtual camera's diaphragm.

Disk (to emulate a perfect circle), Triangle (3 blades), Square (4 blades), Pentagon (5 blades), Hexagon (6 blades), Heptagon (7 blades) or Octagon (8 blades).

Angle

This slider is deactivated, if the Bokeh Type is set to Disk. It can be used to add a rotation offset to the Bokeh shape. The value is the angle in degrees.

Gamma Correction

Applies a gamma correction on the image before and after blurring it.

F-Stop

This option controls the amount of focal blur in the same way as a real camera. It simulates the aperture f of a real lens' iris, without modifying the luminosity of the picture. The default value 128 is assumed to be infinity: everything is in perfect focus. Half the value will double the amount of blur. This slider is deactivated, if *No Z-buffer* is enabled.

Max Blur

This value limits the amount of blur by setting a maximum blur radius. Can be used to optimize the performance. The default value of 0 means no limit.

Threshold *CPU Compositor Only*

Some artifacts, like edge bleed, may occur, if the blur difference between pixels is large. This value controls how large that blur difference considered to be safe.

Tip

Only change this value, if there is an occurring problem with an in-focus object.

Preview *CPU Compositor Only*

If enabled a limited amount of (quasi-)random samples are used to render the preview. This way of sampling introduces additional noise, which will not show up in the final render.

Scene

To select the linked scene.

No Z-buffer

Should be activated for a non Z-buffer in the Z input. No Z-buffer will be enabled automatically whenever a node that is not image based is connected to the Z input.

Z Scale

Only active when *No Z-buffer* is enabled. When *No Z-buffer* is used, the input is used directly to control the blur radius (similar to *F-Stop* when using the Z-buffer). This parameter can be used to scale the range of the Z input.

Outputs

Image

Standard color output.

Examples



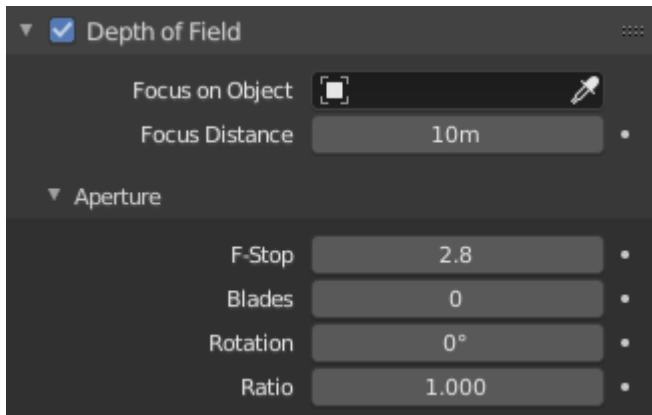
In this [blend-file example](#), the ball array image is blurred as if it was taken by a camera with an f-stop of 2.8 resulting in a fairly narrow depth of field centered on 7.5 units from the camera. As the balls recede into the distance, they get blurrier.

No Z-Buffer Examples

Sometimes might want to have more control to blur the image. For instance, you may want to only blur one object while leaving everything else alone (or the other way around), or you want to blur the whole image uniformly all at once. The node, therefore, allows you to use something other than an actual Z-buffer as the Z input. For instance, you could connect an Image node and use a grayscale image where the color designates how much to blur the image at that point, where white is the maximum blur and black is no blur. Or, you could use a Time node to uniformly blur the image, where the time value controls the maximum blur for that frame. It may also be used to obtain a possibly slightly better DoF blur, by using a fake depth-shaded image instead of a Z-buffer. (A typical method to create the fake depth-shaded image is by using a linear blend texture for all objects in the scene or by using the “fog/mist” fake depth shading method.) This also has the advantage that the fake depth image can have [Anti-Aliasing](#), which is not possible with a real Z-buffer.

The parameter *No Z-buffer*, becomes then the main blur control. The input has to be scaled, because usually the value of a texture is only in the numeric range 0.0 to 1.0.

Camera Settings



Distance setting in the Camera Depth of Field panel.

The *Defocus* node uses the actual camera data in your scene if supplied by a *Render Layer* node.

To set the point of focus, the camera now has a *Distance* parameter, which is shorthand for Depth of Field Distance. Use this camera parameter to set the focal plane of the camera (objects Depth of Field Distance away from the camera are in focus). Set *Distance* in the main *Camera* edit panel; the button is right below the *Depth of Field*.

To make the focal point visible, enable the camera *Limits* option, the focal point is then visible as a yellow cross along the view direction of the camera.

Hints

Preview

In general, use preview mode, change parameters to your liking, only then disable preview mode for the final render.

Edge Artifacts

For minimum artifacts, try to setup your scene such that differences in distances between two objects that may visibly overlap at some point are not too large.

“Focus Pull”

Keep in mind that this is not real DoF, only a post-processing simulation. Some things cannot be done which would be no problem for real DoF at all. A typical example is a scene with some object very close to the camera, and the camera focusing on some point far behind it. In the real world, using shallow depth of field, it is not impossible for nearby objects to become completely invisible, in effect allowing the camera to see behind them. Hollywood cinematographers use this visual characteristic to achieve the popular “focus pull” effect, where the focus shifts from a nearby to a distant object, such that the “other” object all but disappears. Well, this is simply not possible to do with the current post-processing method in a single pass. If you really want to achieve this effect, quite satisfactorily, here is how:

- Split up your scene into “nearby” and “far” objects, and render them in two passes.
- Now, combine the two results, each with their own “defocus” nodes driven by the same Time node, but with one of them inverted (e.g. using a Map Value node with a Size of -1). As the defocus of one increases, the defocus on the other decreases at the same rate, creating a smooth transition.

Aliasing at Low f-Stop Values

At very low values, less than 5, the node will start to remove any oversampling and bring the objects at DoF Distance very sharply into focus. If the object is against a contrasting background, this may lead to visible stair-stepping (aliasing) which OSA is designed to avoid. If you run into this problem:

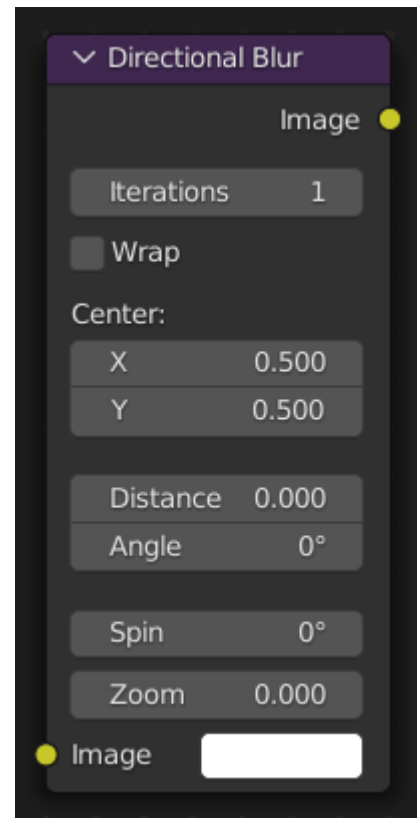
- Do your own OSA by rendering at twice the intended size and then scaling down, so that adjacent pixels are blurred together.
- Use the Blur node with a setting of 2 for X and Y.
- Set DoF Distance off by a little, so that the object in focus is blurred by the tiniest bit.
- Use a higher f-stop, which will start the blur, and then use the Z socket to a Map Value to a Blur node to enhance the blur effect.
- Rearrange the objects in your scene to use a lower-contrast background.

No Z-Buffer

A final word of warning, since there is no way to detect if an actual Z-buffer is connected to the node, be **very** careful with the *No Z-buffer* switch. If the *Z scale* value happens to be large, and you forget to set it back to some low value, the values may suddenly be interpreted as huge blur radius values that will cause processing times to explode.

Directional Blur Node

Blurs an image along a specified direction. Can be used to fake motion blur.



Inputs

Image

Standard color input.

Properties

Iterations

Controls how many times the image is duplicated to create the blur effect. Higher values give smoother results.

Center X, Y

Sets the position where the blur center is. This makes a difference if the angle, spin, and/or zoom are used.

Distance

How large the blur effect is.

Angle

Image is blurred at this angle from the center.

Spin

Rotates the image each iteration to create a spin effect, from the center point.

Zoom

Scales the image each iteration, creating the effect of a zoom.

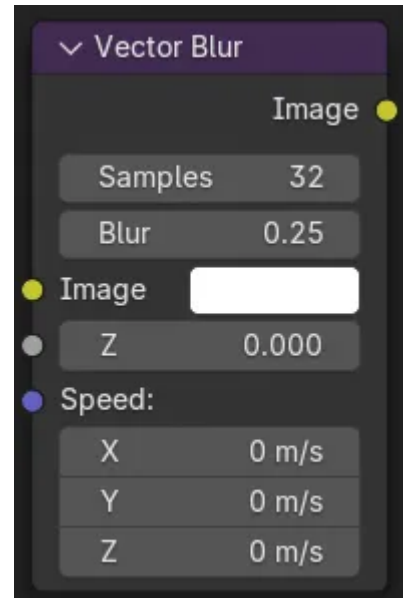
Outputs

Image

Standard color output.

Vector Blur Node

The Vector Blur node is a fast method for simulating [Motion Blur](#) in compositing. It uses the vector speed render pass to blur the image pixels in 2D.



Inputs

Image

Image input, to be linked to the “Combined” render pass.

Z

Z depth, to be linked to the “Depth” render pass.

Speed

Input for the “Vector” render pass. See [Cycles render passes](#).

Properties

Samples

Quality factor.

Blur

Scaling factor for the motion vector (actually the “shutter speed” in frames).

Outputs

Image

Motion blurred image output.

Usage

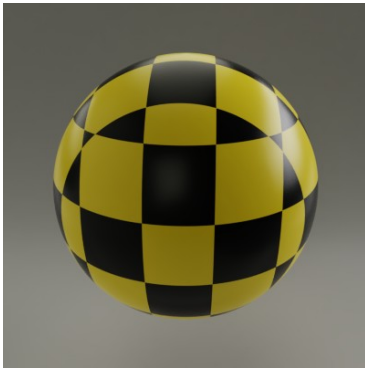
Even with a correct compositing setup with Image, Z and Speed nodes all linked to the appropriate passes, there may still be artifacts. The 2D render passes does not contain 3D information, and so the information what is behind a moving object or outside the camera view is lost.

Better results can be achieved by rendering the scene into multiple render layers, applying vector blur to each render layer, and then compositing the results together. Typically an animated character would be rendered in a separate render layer than the background set. Especially if hair or transparency is involved this is important.

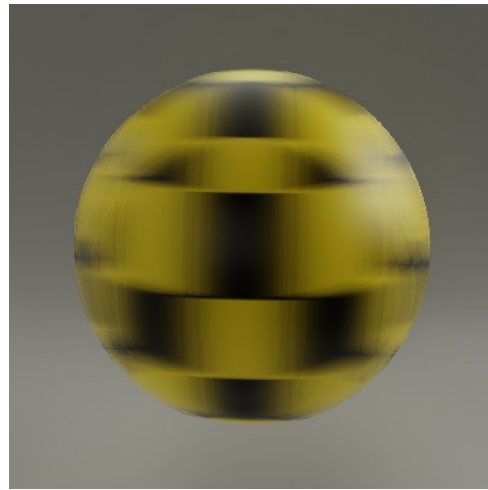
For other artifacts it can help to slightly blur the Speed pass or to set a Maximum Speed limit. This helps to smoothen out the motion, but too much blurring leads to its own problems.

Example

The *speed vector* in this example was created by animating the patterned sphere horizontally and using a frame at the mid-point of the sequence.



Render result, no post-processing.

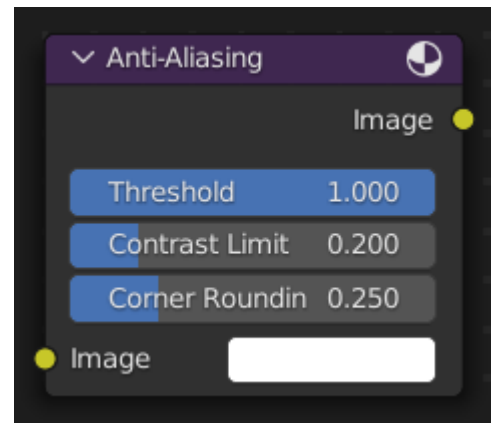


Composite, with Samples set to 32 and Blur set to 1.0.



Anti-Aliasing Node

The Anti-Aliasing node smooths away jagged edges.



Inputs

Image

Standard color input.

Properties

Threshold

Controls edge detection sensitivity across the whole image.

Contrast Limit

Controls contrast level to consider when detecting edges.

The human eye does not perceive all edges equally. For instance, it tends to mask low contrast edges in the presence of much higher contrasts in the surrounding area. Therefore, applying anti-aliasing to unperceived edges will produce artifacts. This option quantifies the difference between low contrast and high contrast neighboring edges.

Corner Rounding

Detect corners to help preserve the original shape. Setting *Corner Rounding* to 0 means no corner detection and no corner rounding will take place. The higher the value the better corners will be preserved, i.e. resemble original image.

Outputs

Image

Standard color output.

Examples

The Anti-Aliasing node has three properties shown here.



Changing
threshold affects
all edges.



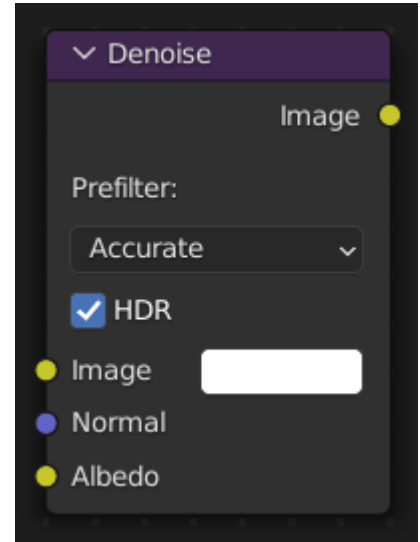
Changing contrast limit affects
neighboring edges below
contrast limit.

The effect of corner rounding.

Denoise Node

The Denoise node is used to denoise renders from [Cycles](#) and other ray tracing renderers. This helps to significantly reduce render time by rendering with fewer samples.

It uses [Open Image Denoise](#), which transforms noisy images into clean images with machine learning.



Inputs

Image

Noisy image input.

Normal

Optional normal render pass to better preserve detail. For Cycles, it is recommended to use the Denoising Normal render pass, which is available when enabling the Denoising Data passes.

Albedo

Optional albedo render pass to better preserve detail. For Cycles, it is recommended to use the Denoising Albedo render pass, which is available when enabling the Denoising Data passes.

Properties

Prefilter

None::	Does not apply any prefiltering to the input passes. This option retains the most detail and is the fastest, but assumes the input passes are noise free which may require a high sample count. If the input passes are not noise free, then noise will remain in the image after denoising.
Fast::	Assumes the input passes are not noise free, yet does not apply prefiltering to the input passes. This option is faster than <i>Accurate</i> but produces a blurrier result.
Accurate::	Prefilters the input passes before denoising to reduce noise. This option usually produces more detailed results than <i>Fast</i> with increased processing time.

HDR

Preserve colors outside the 0 to 1 range.

Outputs

Image

Denoised image output.

Examples

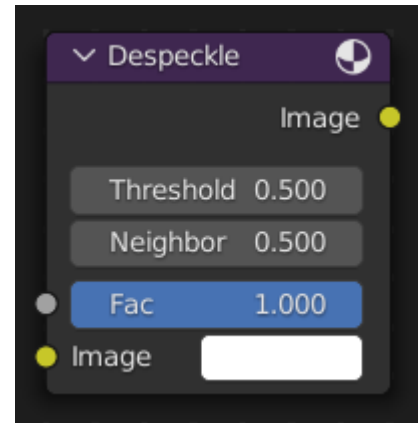


Render before and after denoising, with a very low number of samples as input. As more samples are used, the denoiser will be able to better preserve detail.

Despeckle Node

The *Despeckle node* is used to smooth areas of an image in which noise is noticeable, while leaving complex areas untouched.

This works by the standard deviation of each pixel and its neighbors is calculated to determine if the area is one of high complexity or low complexity. If the complexity is lower than the threshold then the area is smoothed using a simple mean filter.



Inputs

Factor

Controls the amount the filter effects the image.

Image

Standard color input.

Properties

Threshold

The threshold to control high/low complexity.

Neighbor

The threshold to control the number of pixels that must match.

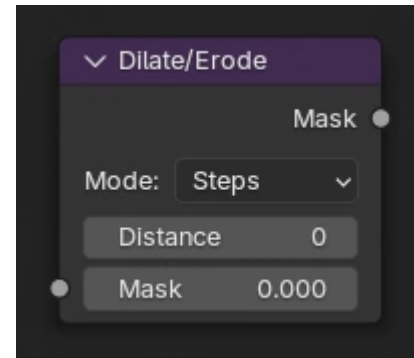
Outputs

Image

Standard color output.

Dilate/Erode Node

Expands or shrinks a mask using a morphological operator.



Inputs

Mask

A grayscale image.

Properties

Mode

Steps

Sets each pixel to the maximum (for dilation) or minimum (for erosion) value that's found within a square surrounding it. This approach keeps the original gray levels and is best suited for masks that contain sharp corners; rounded shapes such as circles will look more square-like in the output.

Despite the name, this is not an iterative process; the dilation/erosion is only performed once regardless of the chosen *Distance*.

Threshold

Makes all the pixels fully black or white depending on whether they're darker or brighter than 50% gray. Then, sets each pixel to the

maximum (for dilation) or minimum (for erosion) value that's found within a circle surrounding it. This approach loses the original gray levels. Shape wise, it's well-suited for masks that contain rounded corners; sharp ones will be rounded off.

Distance

Sets each pixel to the maximum (for dilation) or minimum (for erosion) value that's found within a circle surrounding it. This approach preserves the original gray levels and is well-suited for masks that contain rounded corners.

Feather

Blurs the image.

Distance

The size of the surrounding area to look at for each pixel; or in other words, how much to dilate (for positive values) or erode (for negative values) the mask.

Edge

For the *Threshold* mode, determines how much to blur the edges after dilation/erosion.

Falloff

For the *Feather* mode, determines the brightness curve of the blurred edges.

Outputs

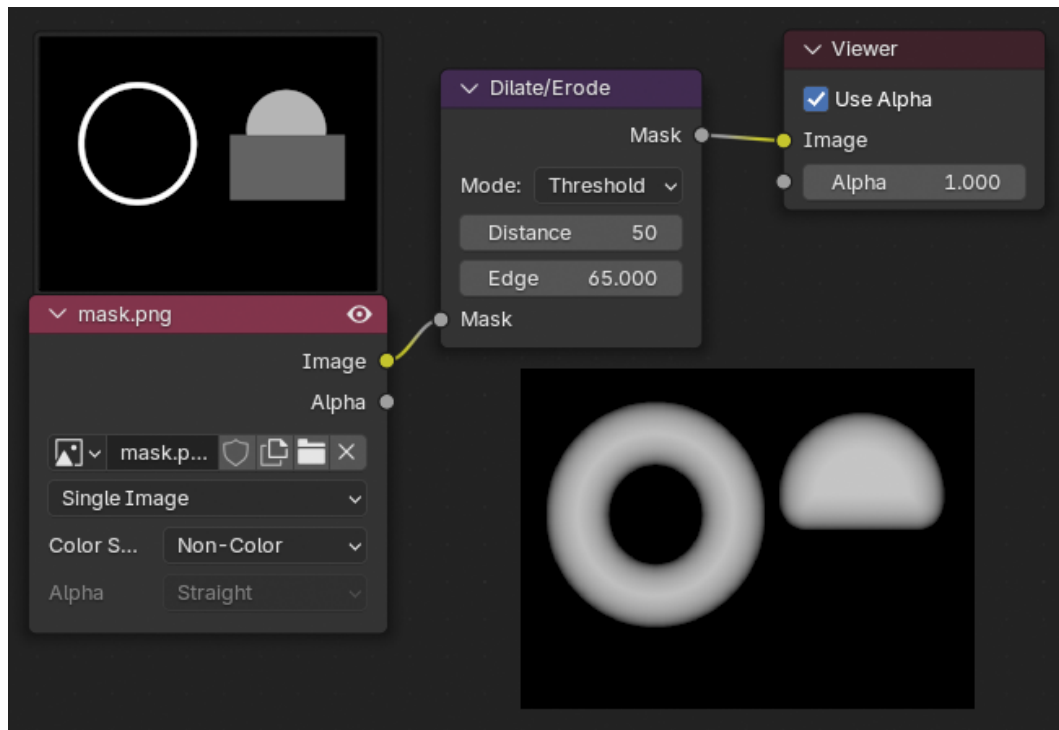
Mask

The resulting mask.

Example

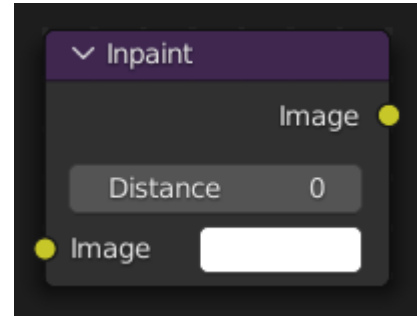
In the image below, notice that:

- The light gray disk has turned white and the dark gray rectangle has turned black because of the *Threshold* mode.
- The shapes have become thicker – dilated because of the positive *Distance*.
- The shapes appear blurred because of the positive *Edge*.



Inpaint Node

The *Inpaint node* is used to extend borders of an image into transparent or masked regions. This can be useful to solve problems like “wire removal” and holes created during chroma keying.



Inputs

Image

Standard color input.

Properties

Distance

The number of times to extend the image.

Outputs

Image

Standard color output.

Examples

The left image shows the “wire” in place and after chroma keying has been applied. You will see you are left with a blank space – it’s shown as a black line here but it will be alpha in your Blender output.



Inpaint Node example.

Inpainting fills in a couple of pixels using the surrounding image and voilà... your wire is removed.

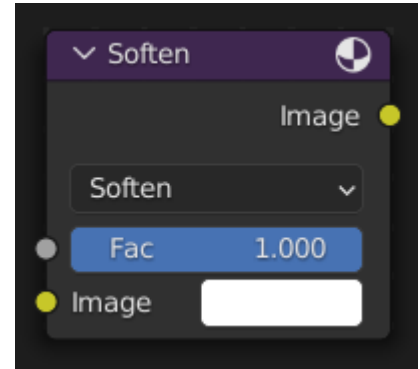
Note

The wider the “hole” is, the more noticeable this effect is! If you use more than a few pixels of infill, the effect is almost as irritating as the wire and your viewers won’t be impressed.

Inpainting can also cover up a multitude of other minor sins such as control points for motion capture: use it sparingly and it will amaze.

Filter Node

The Filter node implements various common image enhancement filters.



Inputs

Factor

Controls the amount of influence the node exerts on the output image.

Image

Standard color input.

Properties

Type

The Soften, Laplace, Sobel, Prewitt and Kirsch all perform edge detection (in slightly different ways) based on vector calculus and set theory equations.

Soften:: Slightly blurs the image.

Box Sharpen:: Increases the contrast, especially at edges.

Diamond Sharpen:: Less aggressive than box sharpen, reducing sharpening artifacts.

Laplace:: Edge highlighting filter susceptible to highlighting visual noise.

Sobel:: Creates a negative image that highlights edges.

Prewitt:: Produces a similar results to Sobel.

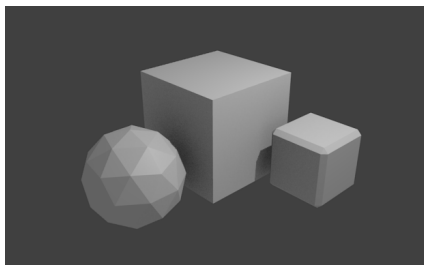
Kirsch::	Gives better blending than Sobel or Prewitt, when approaching an edge.
Shadow::	Performs a relief, emboss effect, darkening outside edges.

Outputs

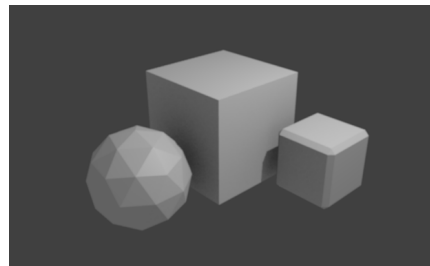
Image
Standard color output.

Example

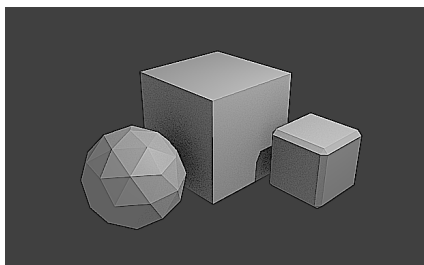
The Filter node has eight modes, shown here.



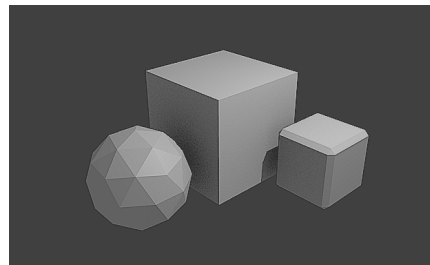
Original image.



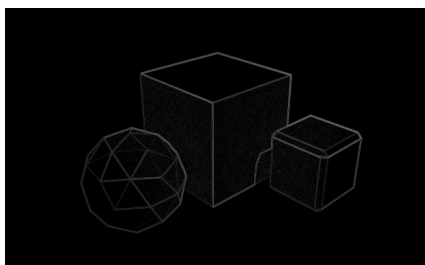
Soften.



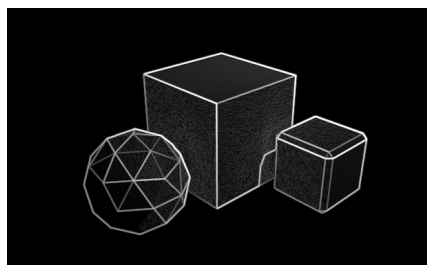
Box Sharpen.



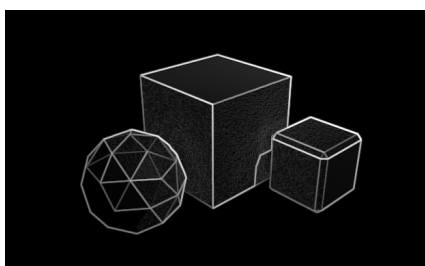
Diamond Sharpen.



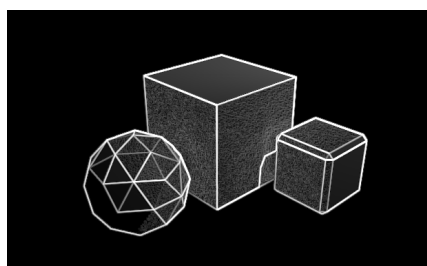
Laplace.



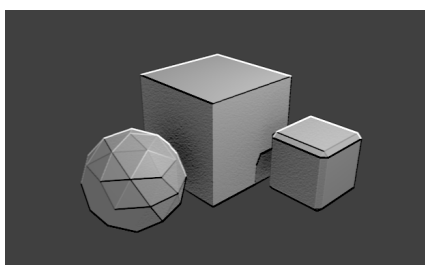
Sobel.



Prewitt.



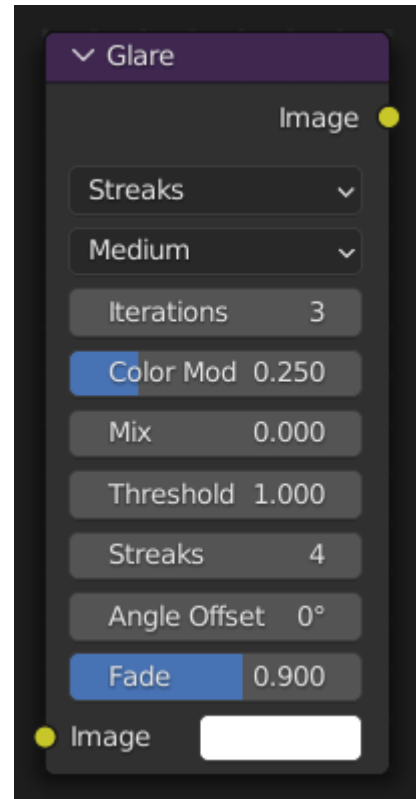
Kirsch.



Shadow.

Glare Node

The *Glare node* is used to add lens flares, fog, glows around bright parts of an image.



Inputs

Image

Standard color input.

Properties

Glare Type

Bloom::

Simulates the glow around bright objects caused by light scattering in eyes and cameras.

Size

Scale of the glow relative to the size of the image. 9 means the glow can cover the entire

image, 8 means it can only cover half the image, 7 means it can only cover quarter of the image, and so on.

Ghosts:: Creates a haze over the image.

Streaks:: Creates bright streaks used to simulate lens flares.
Streaks

Total number of streaks.

Angle Offset

The rotation offset factor of the streaks.

Fade

Fade out factor for the streaks.

Fog Glow:: Simulates the glow around bright objects caused by light scattering in eyes and cameras. This is similar to the *Bloom* mode, but is more physically accurate, at the cost of much slower computation time.

Size

Scale of the glow relative to the size of the image. 9 means the glow will cover the entire image, 8 means it will cover half the image, 7 means it will cover quarter of the image, and so on.

Simple Star:: Works similar to *Streaks* but gives a simpler shape looking like a star.

Fade

Fade out factor for the streaks.

Rotate 45

Rotate the streaks by 45°.

Quality

If not set to something other the *High*, then the glare effect will only be applied to a low resolution copy of the image. This can be helpful to save render times while only doing preview renders.

Iterations

The number of times to run through the filter algorithm. Higher values will give more accurate results but will take longer to compute. Note that, this is not available for *Fog Glow* as it does not use an iterative-based algorithm.

Color Modulation

Used for *Streaks* and *Ghosts* to create a special dispersion effect.

Johannes Itten describes this effect, Color Modulation, as subtle variations in tones and chroma.

Mix

Value to control how much of the effect is added on to the image. A value of -1 would give just the original image, 0 gives a 50/50 mix, and 1 gives just the effect.

Threshold

Pixels brighter than this value will be affected by the glare filter.

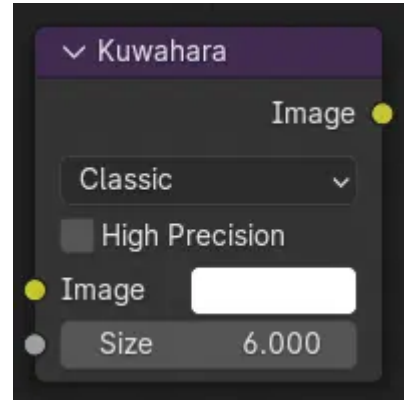
Outputs

Image

Standard color output.

Kuwahara Node

The Kuwahara node implements the Kuwahara filter as well as its anisotropic variant. The Kuwahara filter is a smoothing filter that tries to preserve the edges in the image. The smoothing effect of the anisotropic variant is similar to brush strokes, so the node can be used to create stylized painting effects.



Inputs

Image

Standard color input.

Size

Controls the size of the smoothing neighborhood. Large values may introduce artifacts for highly detailed areas. For the anisotropic method, the larger the size, the slower the filter.



Original.



Size: 3.





Size: 6.



Size: 9.

Properties

Type

Classic::

A simple smoothing method that averages the local square neighborhood of the image while preserving edges. Produces blocky results due to the square neighborhood and provides no tuning parameters, but is faster to compute.

Anisotropic::

A complex smoothing method that averages the local neighborhood of the image in the direction of the flow of the edges, thus preserving the edges in the output. Produces painterly-like results and provides multiple tuning parameters, while being slower to compute.

High Precision

Uses a more precise but slower method. Use if the output contains undesirable noise.

Uniformity

Controls the uniformity of the directions of the edges of the image. Non uniform directions are nearly never desirable, so this should typically be increased until the user notices the result is no longer changing in a significant way. Further increases would produce worst results and increase compute time.

Sharpness

Controls the sharpness of the edges of the image.



Original.



Sharpness: 0.



Sharpness: 0.5.



Sharpness: 1.

Eccentricity

Controls how thin and directional the filter is. Low eccentricity corresponds to circular omnidirectional features while high eccentricity corresponds to thin directional features.



Original.



Eccentricity: 0.



Eccentricity: 1.



Eccentricity: 2.

Outputs

Image

Standard color output.

Notes

Iterations

The filter can be applied multiple times by chaining the node multiple times. This chaining can produce more flat filtering.



Original.



Iterations: 1.



Iterations: 2.



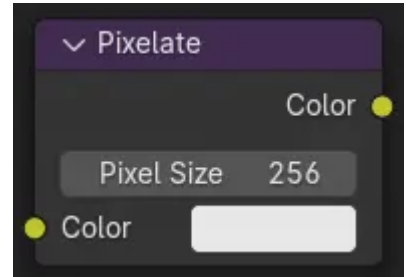
Iterations: 3.

Performance

The filter can be expensive to compute for high size input and high resolution images. To improve performance, consider scaling down the image, applying the filter, then scaling it up again. This can work well because the filter already attenuates low frequency details.

Pixelate Node

The Pixelate node reduces the detail in an image by making individual pixels more prominent. It results in a blocky or mosaic-like appearance, where the fine details of the image are obscured, and the overall image is represented using larger, more noticeable pixels.



Inputs

Color

Standard color input.

Properties

Pixel Size

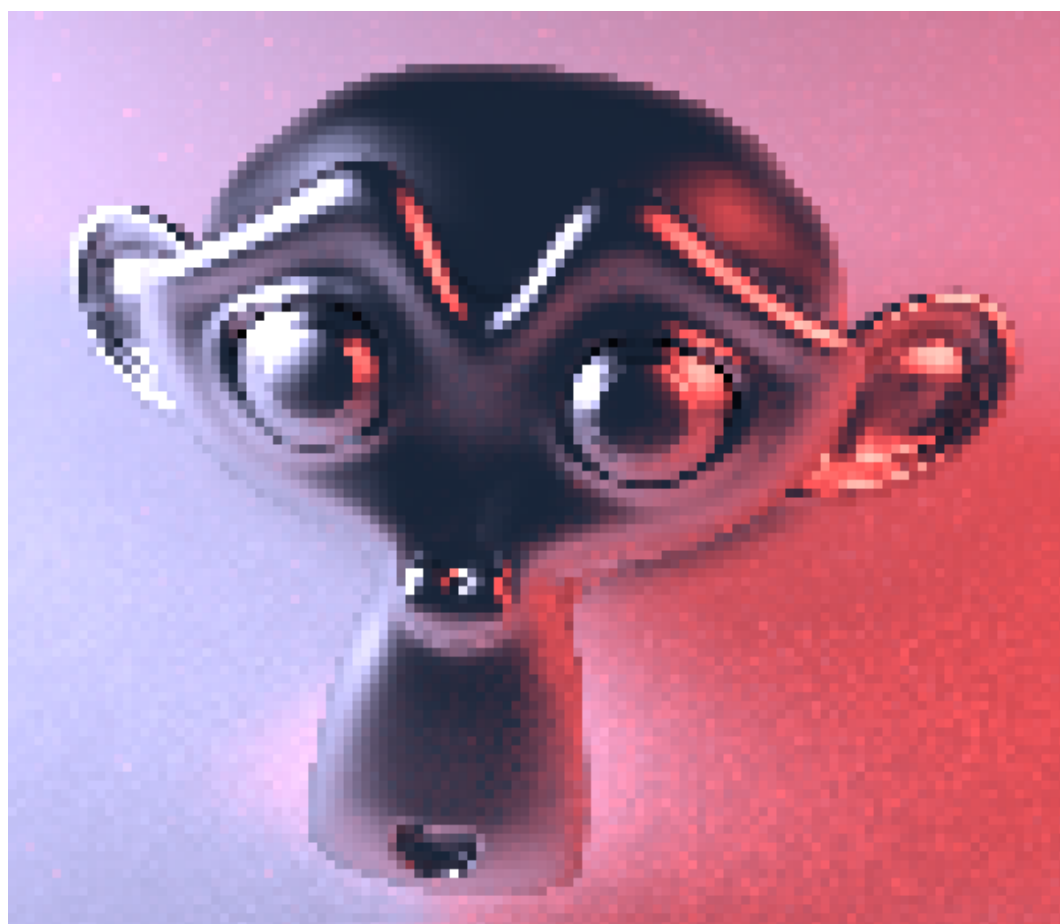
The size of the pixels in the output image, measured in pixels.

Outputs

Color

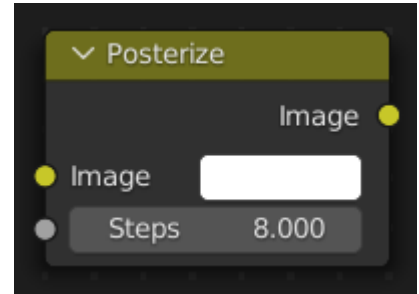
Standard color output.

Example



Posterize

The *Posterize Node* reduces the number of colors in image, converting smooth gradients into sharp transitions. This node is useful for generating masks in particular for rotoscoping.



Inputs

Image

Standard color input.

Steps

The number of colors per channel; A value of 8 will result in $(8^3 = 512)$ total colors.

Properties

This node has no properties.

Outputs

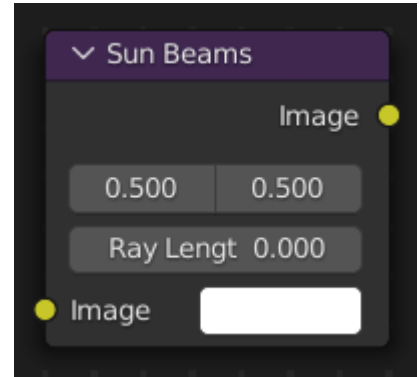
Image

Standard color output.

Sun Beams Node

The Sun Beams node provides a cheap way of adding sun beams based on image brightness alone.

Sun Beams is a 2D effect for simulating the effect of bright light getting scattered in a medium ([Crepuscular Rays](#)). This phenomenon can be created by renderers, but full volumetric lighting is a rather arduous approach and takes a long time to render.



Inputs

Image

Standard color input.

Properties

Source Width, Height

Source point of the rays as a factor of the image dimensions.

Ray Length

Length of the rays as a factor of the image size.

Outputs

Image

Standard color output.

Example

Usually, the first step is to define the area from which rays are cast. Any diffuse reflected light from surfaces is not going to contribute to such scattering in the real world, so should be excluded from the input data. Possible ways to achieve this are:

- Entirely separate image as a light source.
- Brightness/contrast tweaking to leave only the brightest areas.
- Muting shadow and midtone colors, which is a bit more flexible.
- Masking for ultimate control.

After generating the sun beams from such a light source image they can then be overlaid on the original image. Usually, a simple “Add” Mix node is sufficient, and physically correct because the scattered light adds to the final result.



Keying Nodes

These nodes give you the essential tools for creating a [Matte](#) for images that do not already have their own [Alpha Channel](#). One usage scenario is blue-screen or green-screen footage, where live action is shot in front of a blue or green backdrop for replacement by a matte painting or virtual background.

In general, hook up these nodes to a viewer, set your Image Editor to show the Viewer node, and play with the sliders in real-time using a sample image from the footage, to get the settings right. In some cases, small adjustments can eliminate artifacts or foreground image degradation. Taking out too much green can result in foreground actors looking flat or bluish/purplish.

You can and should chain these nodes together, improving your masking and color correction in successive refinements, using each node's strengths to operate on the previous node's output. [Keying Node](#) is the closest to a "does-it-all" node for green screens, but the best results stem from a combination of techniques.

Note

Garbage Matte is not a node, but a technique selecting what to exclude from an image. It is a [Mask](#) used to identify content to be removed from an image that cannot be removed by an automatic process like chroma keying. It is used either to select specific content to be removed, or it is the inverse of a rough selection of the subject; removing everything else.

Some nodes accept a garbage matte directly. For those that don't, you can still apply one by subtracting the garbage matte from the matte generated by the node.

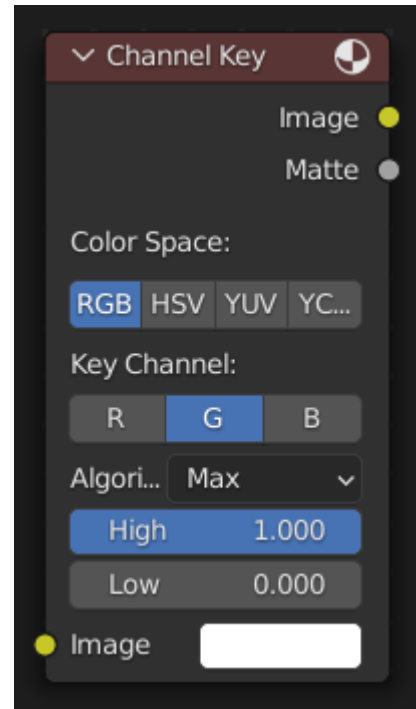
Simple garbage mattes can be created with the [Box Mask](#) or the [Ellipse Mask](#). More complicated matte shapes using a [Double Edge Mask](#) or using a [Mask](#).

- [Channel Key Node](#)
- [Chroma Key Node](#)
- [Color Key Node](#)
- [Color Spill Node](#)
- [Difference Key Node](#)
- [Distance Key Node](#)
- [Keying Node](#)
- [Keying Screen Node](#)
- [Luminance Key Node](#)

Channel Key Node

The *Channel Key* node determines background objects from foreground objects by the difference in the selected channel's levels.

For example in the YUV [Color Model](#), this node is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.



Inputs

Image

Standard color input.

Properties

Color Space

This button selects what color model the channels will represent.

RGB, HSV, YUV, YCbCr

Key Channel

This button selects the channel, defined by the *Color Space*, to use to determine the matte.

Algorithm

Method to calculate the difference between levels.

Max:: Limit by the maximum of the other two channels other than the *Key Channel*.

Single:: Limit by the maximum of the selected *Limiting Channel*.

Limiting Channel

The channel to use when computing the maximum, the options are defined by the *Color Space*.

High

Determines the lowest values that are considered foreground. (Which is supposed to be – relatively – high values: from this value to 1.0.)

Low

Determines the highest values that are considered to be background objects. (Which is supposed to be – relatively – low values: from 0.0 to this value.)

Tip

It is possible to have a separation between the *High* and *Low* values to allow for a gradient of transparency between foreground and background objects.

Outputs

Image

Image with an alpha channel adjusted for the keyed selection.

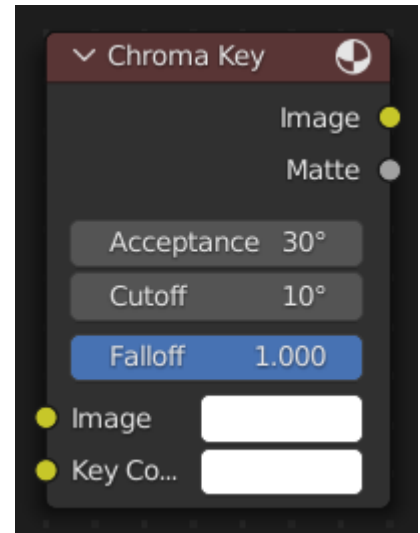
Matte

A black-and-white alpha mask of the key.

Chroma Key Node

The *Chroma Key* node determines if a pixel is a foreground or background (and thereby should be transparent) based on its chroma values.

Use this, for example, to composite images that have been shot in front of a green or blue screen.



Inputs

Image

Standard color input.

Key Color

The background color usually selected using the color picker and the original image.

Properties

Acceptance

An angle on the color wheel that represents how tolerant the keying color is. Larger angles allow for larger variation in the keying color to be considered background pixels.

Cutoff

Controls the level that is considered the pure background. Higher cutoff levels mean more pixels will be 100% transparent if they are within the

angle tolerance.

Falloff

Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

Outputs

Image

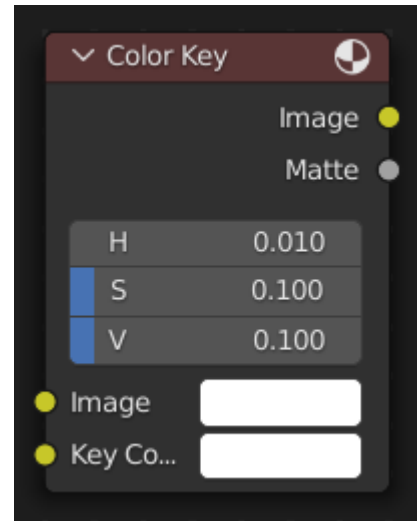
Image with its alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Color Key Node

The Color Key node creates a matte based on a specified color of the input image.



Inputs

Image
Standard color input.

Properties

Color
The sliders represent threshold values. Higher values in this node's context mean a wider range of colors from the specified will be added to the matte.

Hue, Saturation, Value

Outputs

Image
Image with its alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Color Spill Node

The *Color Spill* node reduces one of the RGB channels so that it is not greater than any of the others.

This is common when compositing images that were shot in front of a green or blue screen. In some cases, if the foreground object is reflective, it will show the green or blue color; that color has “spilled” onto the foreground object. If there is light from the side or back, and the foreground actor is wearing white, it is possible to get “spill” green (or blue) light from the background onto the foreground objects, coloring them with a tinge of green or blue. To remove the green (or blue) light, you use this fancy node.



Inputs

Image

Standard color input.

Factor

Standard Factor.

Properties

Despill Channel

R, G, B

Algorithm

Simple, Average

Limiting Channel

R, G, B

Ratio

Scale limit by value.

Unspill

Allows you to reduce the selected channel's input to the image greater than the color spill algorithm normally allows. This is useful for exceptionally high amounts of the color spill.

R, G, B

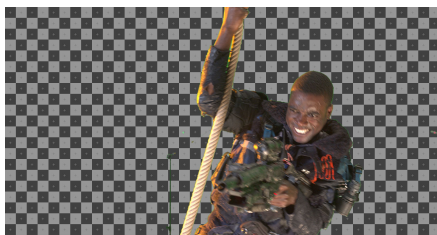
Outputs

Image

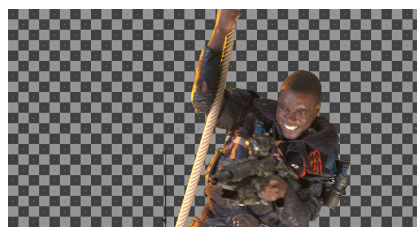
The image with the corrected channels.

Example

Results with the nodes applied to an image from the [Mango Open Movie](#).



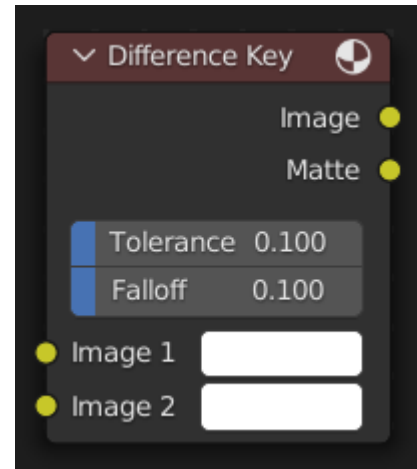
Before: green border and green reflections.



After: no unwanted green.

Difference Key Node

This node produces a matte that isolates foreground content by comparing it with a reference background image.



Inputs

Image

Contains foreground content against the background that is to be removed.

Image

The reference background image.

Properties

Tolerance

Where pixels match the reference background to within the specified threshold, the matte is made transparent.

Falloff

Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

Outputs

Image

Image with its alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Distance Key Node

The Distance Key node determines a pixel's alpha value based on the three-dimensional distance between the image pixel color and the key color in a 3D color space.

This key works well when trying to single out a specific color in a background (not necessarily green).



Inputs

Image

Standard color input.

Key Color

The color that is to be keyed.

Properties

Tolerance

A threshold what the node considers a match between the key color and the foreground pixel. The tolerance affects how close a pixel needs to be to the background pixel to be considered an absolute match.

Falloff

When the Falloff value is high, pixels that are close to the Key Color are more transparent than pixels that are not as close to the Key Color (but

still considered close enough to be keyed). When the Falloff value is low, it does not matter how close the pixel color (Image) is to the Key Color, it is transparent.

Color Space

It is also possible to work with YCbCr color space, but only the Cb and Cr channels are taken into consideration for determining the distance between the foreground and background pixels.

RGB, YCC

Outputs

Image

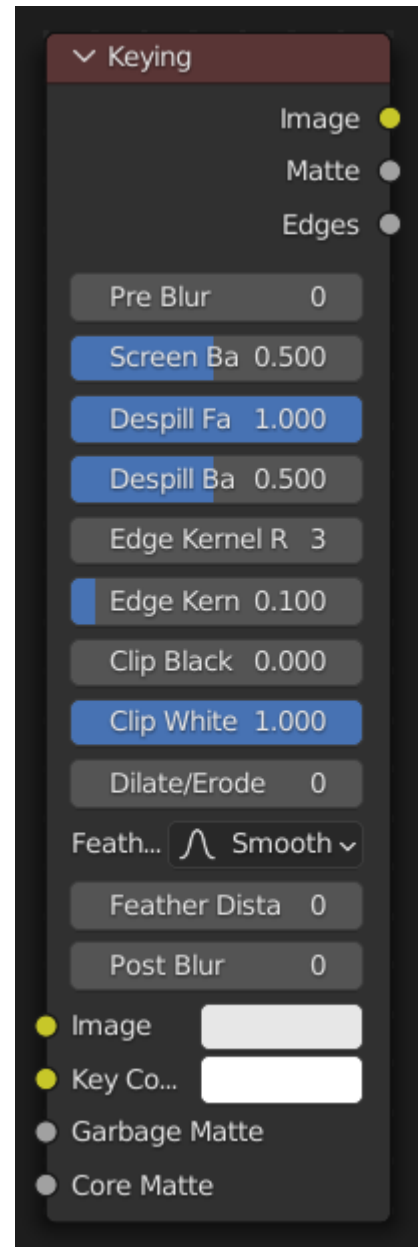
The image with an alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Keying Node

The *Keying* node is a one-stop-shop for “green screen” / “blue screen” removal. It performs both chroma keying to remove the backdrop and despill to correct color cast from the backdrop. Additionally, you can perform common operations used to tweak the resulting matte.



Inputs

Image

Standard color input.

Key Color

The color of content to be removed. This may be a single color, or a reference image such as generated by the [Keying Screen Node](#).

Garbage Matte

An optional mask of area(s) to always *exclude* from the output. This is removed from the chroma key generated matte.

Core Matte

An optional mask of area(s) to always *include* in the output. This is merged with the chroma key generated matte.

Properties

Pre Blur

Reduce the effects of color noise in the image by blurring only color by the given amount, leaving luminosity intact. This will affect matte calculation only, not the result image.

Screen Balance

This is the balance between color channels compared with the key color. 0.5 will average the other channels (red and blue in the case of a green screen).

This may be tweaked in tandem with *Clip Black* and *Clip White* while checking the *Matte* output to create a mask with optimal separation.

Despill Factor

Controls how much color bleed from the key color is removed from the input image: 0 means no despill, 1 means all possible spilling will be removed. The underlying implementation is the same as adjusting the *Unspill* amount of the [Color Spill Node](#).

Despill Balance

This controls how the color channels are compared when computing spill, affecting the hue and shade of the corrected colors. It is similar to setting the *Limiting Channel* in the [Color Spill Node](#).

Edge Kernel Radius

Defines the radius in pixel used to detect an edge.

Edge Kernel Tolerance

Defines threshold used to check if pixels in radius are the same as current pixel: if the difference between pixel colors is higher than this threshold then the point will be considered an edge.

Clip Black

This sets the threshold for what becomes fully transparent in the output (black in the matte). It should be set as low as possible. Uneven backdrops will require this value to be increased. Use of the [Keying Screen Node](#) can help keep this value low. You may also use a *Garbage Matte* to exclude problematic areas.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Clip White

This sets the threshold for what becomes fully opaque in the output (white in the matte). It should be set as high as possible. Colors close to green in the foreground may require reducing this value and/or adjusting the *Screen Balance*. Particularly problematic parts can be fixed with a *Core Matte* instead of a low *Clip White*.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Dilate/Erode

Enlarge (positive numbers) or shrink (negative numbers) the matte by the specified number of pixels. This is similar to using the [Dilate/Erode Node](#) on the matte.

This is a simple way to include more or less along the edges of the matte, particularly combined with *Post Blur*.

Feather Falloff

The rate of the falloff at the edges of the matte when feathering, to manage edge detail.

Feather Distance

Controls how much the matte is feathered inwards (negative number) or outwards (positive number).

Post Blur

Make the matte less sharp, for smoother transitions to the background and noise reduction.

Outputs

Image

Processed image with the *Matte* applied to the images' [Alpha Channel](#).

Matte

Output matte to use for checking the quality of the key, or to manually apply using a [Set Alpha Node](#) or [Mix Node](#).

Edges

Shows what edges were detected on the matte. Useful for adjusting the *Edge Kernel Radius* and *Edge Kernel Tolerance*.

Tip

If there are problems with the edges of the matte, it may help to start with adjusting the *Edge Kernel* parameters before adjusting feathering.

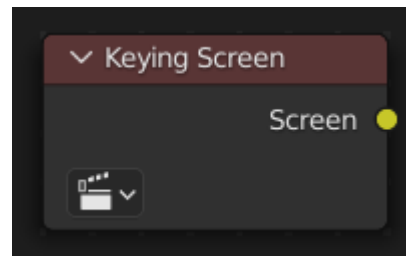
Detected edges are not subject to *Clip Black* / *Clip White* thresholds to preserve fine edge detail. You can check edge detection by connecting a [Viewer Node](#) to the *Edges* output.

Sharper detected edges (smaller *Edge Kernel Radius*, like 2 / larger *Edge Kernel Tolerance*, like 0.4) will create a sharper matte, but may lose some detail like stray hairs. A sharp matte is good, but disappearing or flickering hairs are distracting.

Fat edges (larger *Edge Kernel Radius*, like 8 / smaller *Edge Kernel Tolerance*, like 0.05) will capture more edge detail, but may also produce a halo around the subject. The halo can be adjusted with *Feather* controls along with *Dilate/Erode*.

Keying Screen Node

The Keying Screen node creates plates for use as a color reference for keying nodes. It generates gradients from sampled colors on motion tracking points on movie clips. It can be used to deal with uneven colors of green screens.



Inputs

This node has no input sockets.

Properties

Movie Clip

The selectable clip data-block used as input for the gradient colors.

Tracking Object

Tracking Object to generate the gradient. You will probably want to create new a tracking object in the [Object](#) panel, because tracks used for gradients can not actually be used for camera/object tracking. After this tracks might be placed in places where gradient colors should be sampled. These tracks could be tracked or moved manually, so gradients would be updating automatically along the movie. Tracks might have an offset for easier tracking of feature-less screens.

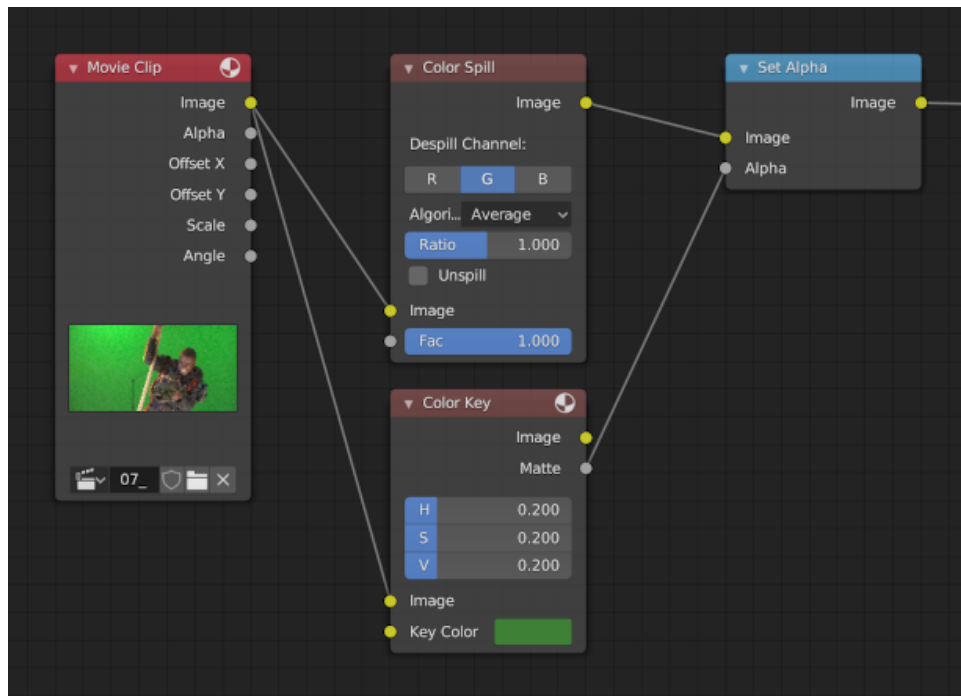
Outputs

Screen

Gradient image output.

Example

Consider a node setup for green screen removal, using a [Color Key](#):



Often, lighting is uneven across the backdrop.



Example from the [Mango Open Movie](#), Tears of Steel.

That can result in a bad matte.

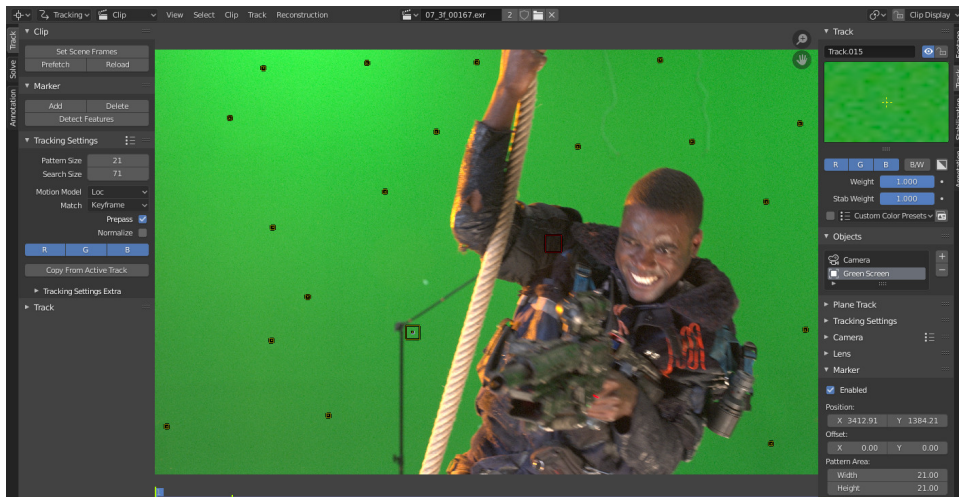


Example of a poor mask: Some of the backdrop is opaque, and some parts of the gun in the foreground are transparent.

If you increase the tolerances on the Color Key node, it will accept more shades of green to mask out. But it may also incorrectly mask out more of the foreground.

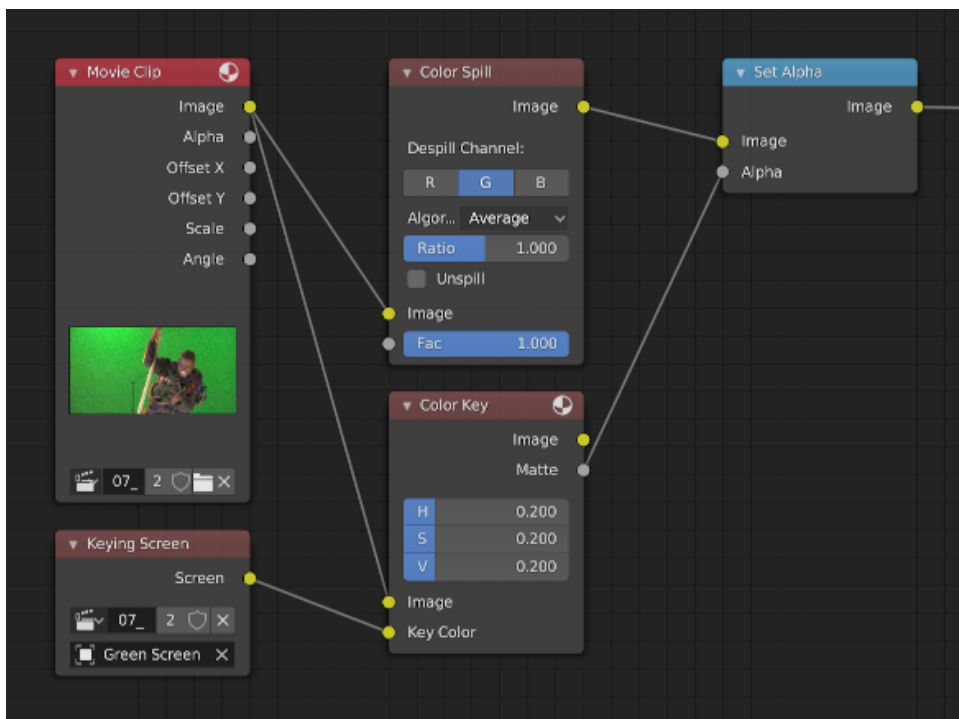
Instead of increasing the range of accepted shades to be masked out, the Keying Screen node lets you change what shade of green (or other color) to used for different parts of the image.

Start in the [Movie Clip Editor](#). Open the Sidebar region and Toolbar to show tracking configuration. Tracks used for gradients are not useful for camera solving, because they do not track well. So create a new object track in the *Objects* selector. Place tracking markers on the clip to sample different parts of the backdrop.



These tracks may be tracked or moved manually, so gradients can be updated over time. If the marker is not enabled for a frame, it will not be used creating the gradient. (Such as the red-colored marker on the arm in the screenshot above)

Once the tracks are created, add the node to your compositing setup, and select the tracking object used for the backdrop.

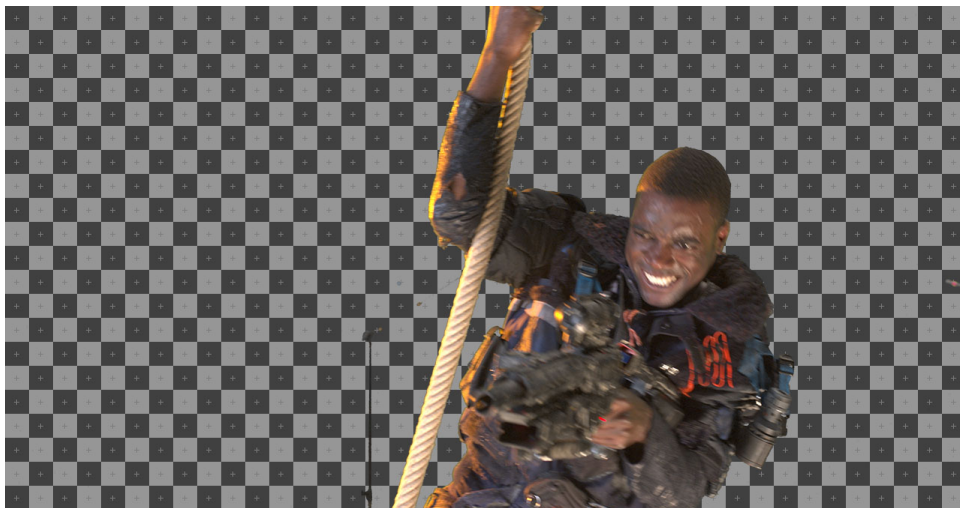


Node configuration with *Keying Screen*'s generated gradient plate connected to the Color input of the Keying node.



Gradient plate generated by *Keying Screen*.

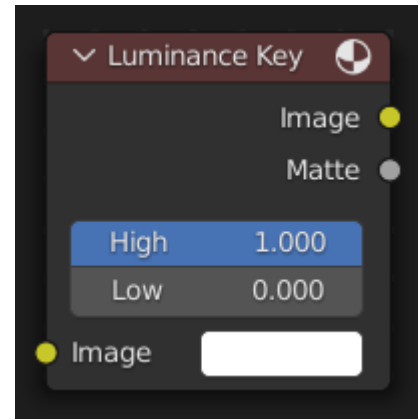
The resulting image now has a better matte.



Luminance Key Node

The *Luminance Key* node determines background objects from foreground objects by the difference in the luminance (brightness) levels.

Stock footage of explosions, smoke or debris are normally shot against a solid, dark background rather than a green screen. This node can separate the foreground effect from the background. It can also be used for sky replacement for overexposed or gray skies that aren't suitable for chroma keying.



Tip

When compositing footage of something that emits light and has a dark background, like fire, a [Mix Node](#) using a *Screen* or *Add* operator will produce better results.

Inputs

Image

Standard color input.

Properties

Limit

High

Determines the lowest values that are considered foreground.
(Which is supposed to be – relatively – light: from this value to 1.0.)

Low

Determines the highest values that are considered to be background objects. (Which is supposed to be – relatively – dark: from 0.0 to this value.)

Note

Brightness levels between the two values form a gradient of transparency between foreground and background objects.

Outputs

Image

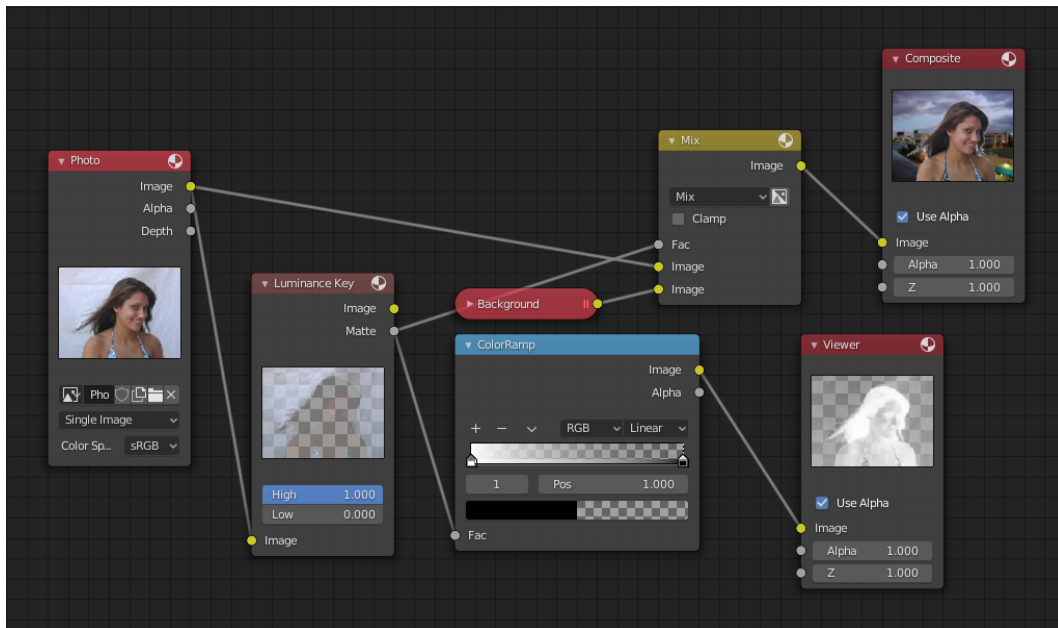
Image with an alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Example

For this example the model was shot against a *white* background. Using the Luminance Key node, we get a matte out where the background is white, and the model is black; the opposite of what we want. If we wanted to use the matte, we have to switch the white and the black. How to do this? Color Ramp node to the rescue – we set the left color to White Alpha 1.0, and the right color to be Black Alpha 0.0. Thus, when the Color Ramp gets in black, it spits out white, and vice versa. The reversed mask is shown; its white outline is usable as an alpha mask now.



Using Luma Key with a twist.

Now to mix, we do not really need the *Alpha Over* node; we can just use the mask as our Factor input. In this kinda weird case, we can use the matte directly; we just switch the input nodes. As you can see, since the matte is white (1.0) where we do not want to use the model picture, we feed the background photo to the bottom socket (recall the Mix node uses the top socket where the factor is 0.0, and the bottom socket where the factor is 1.0). Feeding our original photo into the top socket means it will be used where the Luminance Key node has spit out Black. Voilà, our model is teleported from Atlanta to aboard a cruise ship docked in Miami.

Mask Nodes

- [Cryptomatte Node](#)
 - [Cryptomatte Node \(Legacy\)](#)
-

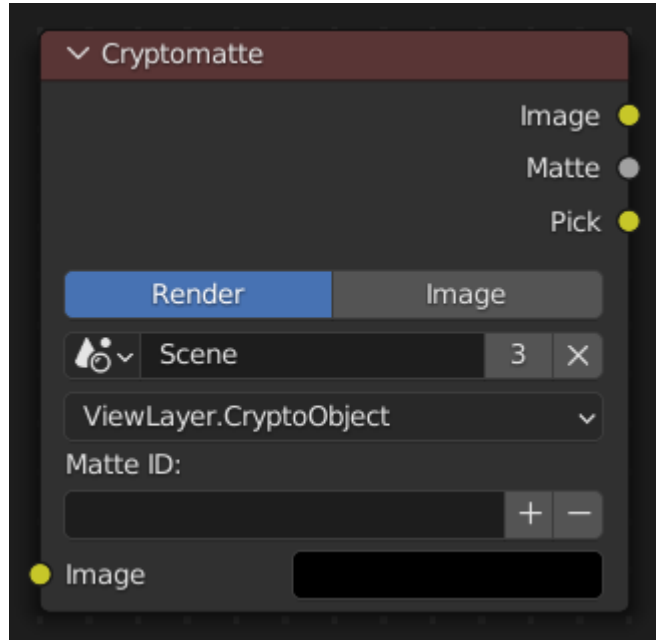
- [Box Mask Node](#)
 - [Ellipse Mask Node](#)
-

- [Double Edge Mask Node](#)
- [ID Mask Node](#)

Cryptomatte Node

The Cryptomatte node uses the Cryptomatte standard to efficiently create mattes for compositing. Cycles and Eevee output the required render passes, which can then be used in the Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing, and mattes will be anti-aliased and take into account effects like motion blur and transparency.



Inputs

Image

Standard color input.

Properties

Source

The source of the Cryptomatte data.

Render::

Use Cryptomatte data that are stored as part of the render.

Image::

Use Cryptomatte data that are stored inside a multilayered OpenEXR image.

Scene

Scene selector. Only available when Render Source is selected.

Image

Image selector. Only available when Image Source is selected.

Cryptomatte Layer

Selector of the Cryptomatte layer.

Matte ID

List of object and material crypto IDs to include in matte. This can be used for example to quickly clear all mattes by deleting the text or used to copy-and-paste crypto IDs from other software.

Outputs

Image

A colored output of the input image with the matte applied to only include selected layers.

Matte

A black-and-white alpha mask of all the selected crypto layers.

Pick

A colored representation of the Cryptomatte pass which can be used as a higher contrast image for color picking.

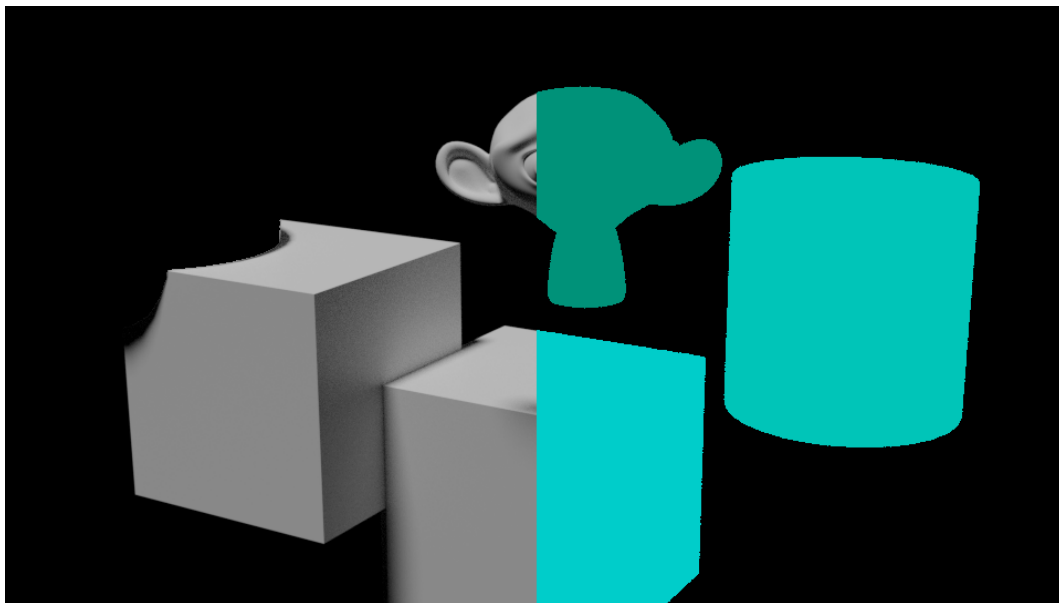
Usage

1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and select the Cryptomatte layer.
3. Attach a Viewer node to the combined pass of the render layers.
4. Use the Cryptomatte Add/Remove button to sample objects from the Compositor backdrop.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

The Image editor, UV editor, node backdrop or Movie Clip editor can be used to pick a Cryptomatte sample. They don't need to show any Cryptomatte layer. The node will use the sample image coordinate to sample in the Cryptomatte layer that is selected in the node.

Example

In the example below, you can see the pass output on the right side. On the left side you can see a couple of objects that were selected through the *Cryptomatte* node. Notice how the cube on the left has a sphere-shaped cut-out from a sphere that was not selected in the node.



Limitations

- Cryptomatte sidecars (metadata files) are not supported.
- Cryptomatte node cannot be used in node groups.
- [Volume Objects](#) are not supported.

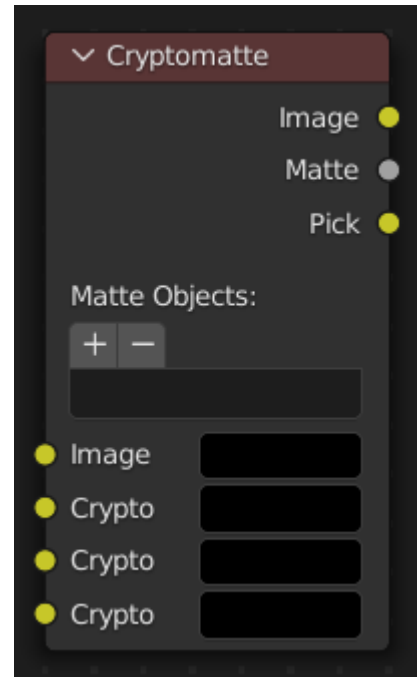
Cryptomatte Node (Legacy)

The Cryptomatte node uses the Cryptomatte standard to efficiently create mattes for compositing. Cycles and Eevee output the required render passes, which can then be used in the Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing, and mattes will be anti-aliased and take into account effects like motion blur and transparency.

Important

The Cryptomatte Legacy node is deprecated and replaced by [Cryptomatte Node](#). The legacy node will be removed in a future Blender release.



Inputs

Image

Standard color input.

Crypto Passes

Each crypto layer will be given its own render pass; each of these render passes must be connected to one of these crypto layer inputs. By default there are only four layers, see [Adding/Removing Layers](#) to add more.

Properties

Add/Remove

Adds/Removes an object or material from matte, by picking a color from the *Pick* output.

Matte ID

List of object and material crypto IDs to include in matte. This can be used for example to quickly clear all mattes by deleting the text or used to copy-and-paste crypto IDs from other software.

Outputs

Image

A colored output of the input image with the matte applied to only include selected layers.

Matte

A black-and-white alpha mask of the all the selected crypto layers.

Pick

A colored representation of the Cryptomatte pass which can be used with a Viewer node to select which crypto passes are used to create the matte image.

Usage

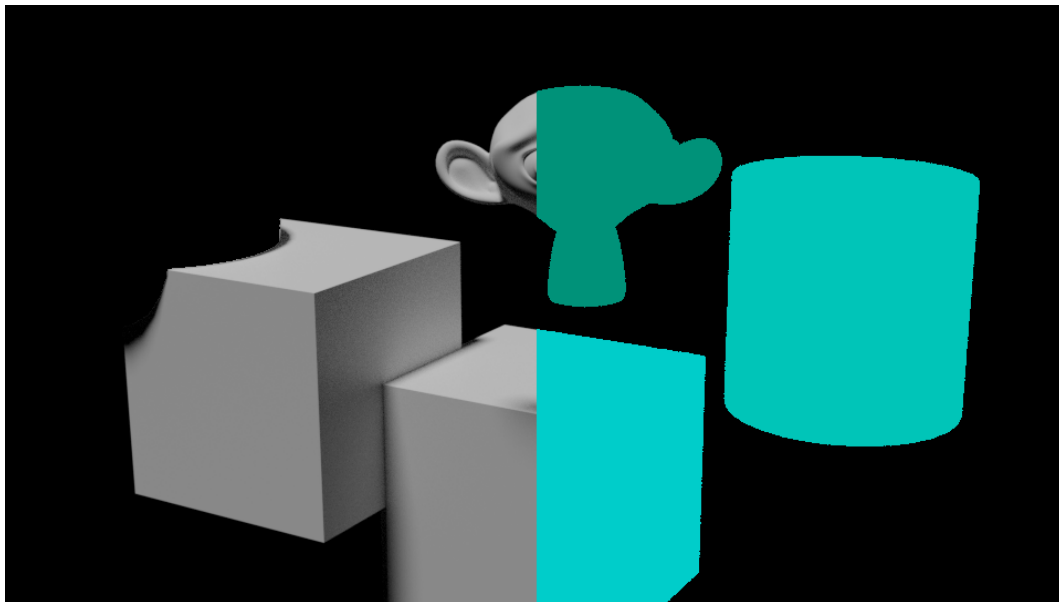
1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and link the Render Layer matching Image and Cryptomatte passes to it.
3. Attach a Viewer node to the Pick output of the Cryptomatte node.
4. Use the Cryptomatte Add/Remove button to sample objects in the Pick Viewer node.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

Adding/Removing Layers

By default there are only four crypto layers available as inputs to the Cryptomatte node. You can add or remove layer inputs through *Sidebar ▶ Item ▶ Properties ▶ Add/Remove Crypto Layer*. These operators will add/remove layers from the bottom of the pass inputs.

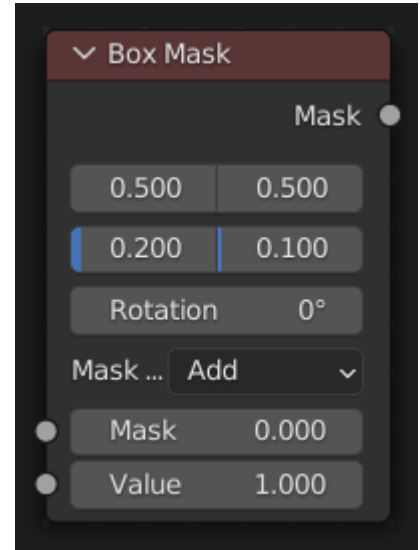
Example

In the example below, you can see the pass output on the right side. On the left side you can see a couple of objects that were selected through the *Cryptomatte* node. Notice how the cube on the left has a sphere-shaped cut-out from a sphere that was not selected in the node.



Box Mask Node

The *Box Mask* node creates an image suitable for use as a simple matte.



Inputs

Mask

An optional mask to use as the base for mask operations.

Value

Intensity of the generated mask.

Properties

X, Y

Position of the center of the box as a fraction of the total width or height. (0.5, 0.5 creates a centered box; 0.0, 0.0 creates a box in the lower left.)

Width

Width of the box as a fraction of the total image width.

Height

Height of the box as a fraction of the total image *width*, not height.

Rotation

Rotation of the box around its center point.

Mask Type

Operation to use against the input mask.

Add::	This yields the <i>union</i> of the input mask and the generated mask: Areas covered by the generated mask are set to the specified <i>Value</i> . Other parts of the input mask are passed through unchanged, or set to black if there is no input mask.
Subtract::	Values of the input mask have the specified <i>Value</i> subtracted from them.
Multiply::	This yields the <i>intersection</i> of this generated mask and the input mask: Values of the input mask are multiplied by the specified <i>Value</i> for the area covered by the generated mask. All other areas become black.
Not::	Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified <i>Value</i> . Areas uncovered by the generated mask remain unchanged.

Outputs

Mask

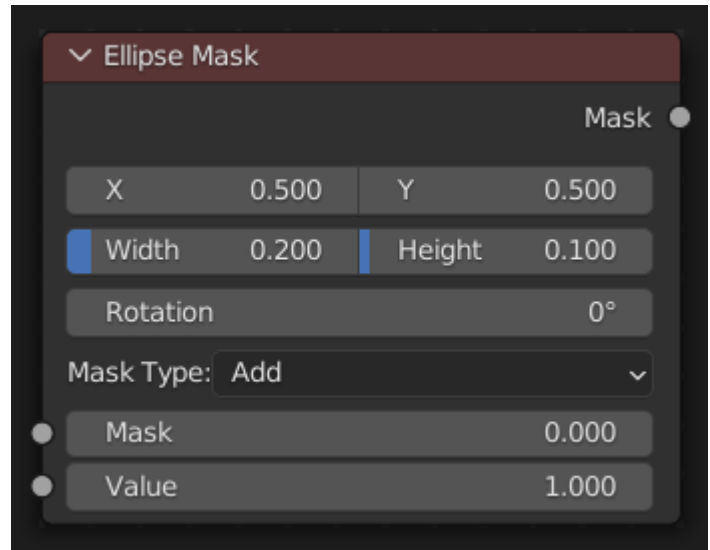
A generated rectangular mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip

For soft edges, pass the output mask through a slight [Blur node](#).

Ellipse Mask Node

The *Ellipse Mask* node creates an image suitable for use as a simple matte or vignette mask.



Inputs

Mask

An optional mask to use as the base for mask operations.

Value

Intensity of the generated mask.

Properties

X, Y

Position of the center of the ellipse as a fraction of the total width or height. (0.5, 0.5 creates a centered ellipse; 0.0, 0.0 creates an ellipse with its center in the lower left.)

Width

Width of the ellipse as a fraction of the total image width.

Height

Height of the ellipse as a fraction of the total image *width*, not height.
Equal *Width* and *Height* values will produce a circle.

Rotation

Rotation of the ellipse around its center point.

Mask Type

Operation to use against the input mask.

- | | |
|-------------------|---|
| Add:: | This yields the <i>union</i> of the input mask and the generated mask: Areas covered by the generated mask are set to the specified <i>Value</i> . Other parts of the input mask are passed through unchanged, or set to black if there is no input mask. |
| Subtract:: | Values of the input mask have the specified <i>Value</i> subtracted from them. |
| Multiply:: | This yields the <i>intersection</i> of this generated mask and the input mask: Values of the input mask are multiplied by the specified <i>Value</i> for the area covered by the generated mask. All other areas become black. |
| Not:: | Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified <i>Value</i> . Areas uncovered by the generated mask remain unchanged. |

Outputs

Mask

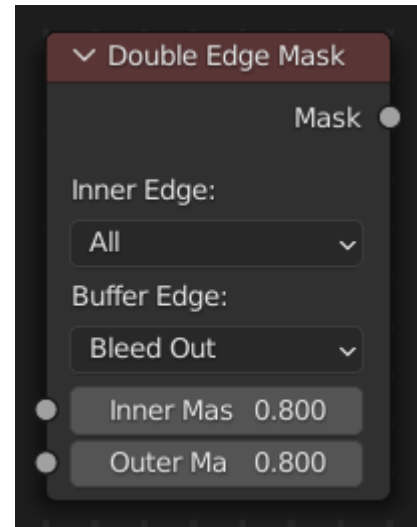
A generated elliptical mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip

For soft edges, pass the output mask through a slight [Blur node](#). For a vignette, pass the output of this through a heavy blur.

Double Edge Mask Node

The *Double Edge Mask* node creates a gradient between two masks.



Inputs

Inner Mask

A mask representing the inside shape, which will be fully white.

Outer Mask

A mask representing the outside shape, which will fade from black at its edges to white at the *Inner Mask*.

Properties

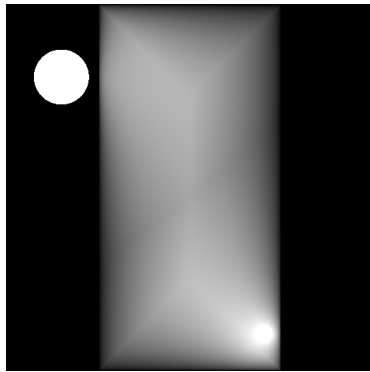
Inner Edge

All::

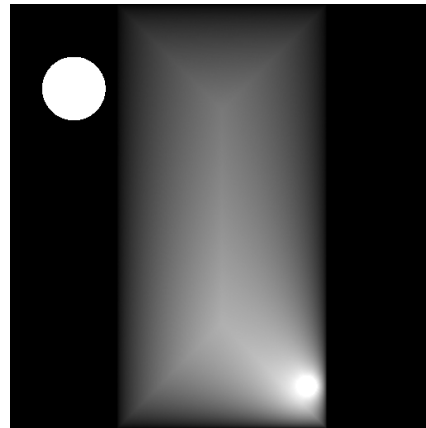
All shapes in the *Inner Mask* contribute to the gradient, even ones that do not touch the *Outer Mask* shape.

Adjacent Only::

Only shapes in the *Inner Mask* that overlap with the *Outer Mask* contribute to the gradient.



All.



Adjacent Only.

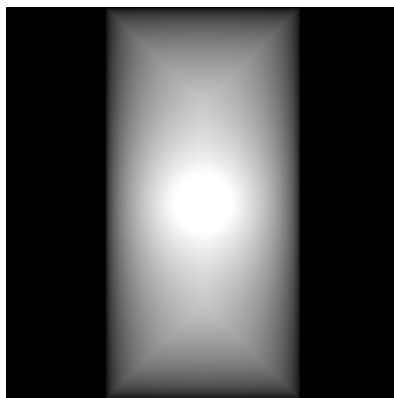
Buffer Edge

Keep In::

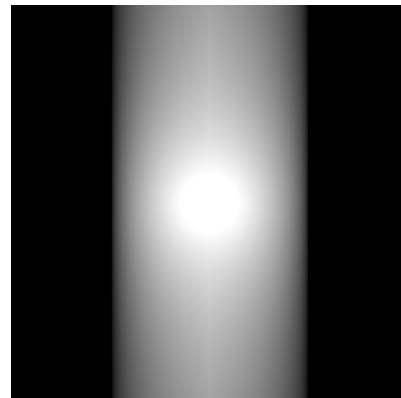
Parts of the *Outer Mask* that touch the edge of the image are treated as if they stop at the edge.

Bleed Out::

Parts of the *Outer Mask* that touch the edge of the image are extended beyond the boundary of the image.



Keep In.



Bleed Out.

Outputs

Mask

Standard mask output.

Example

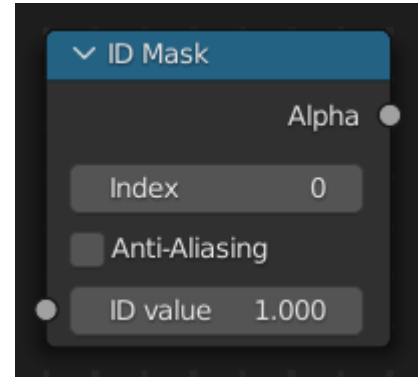
[Double Edge Mask Example Video](#)

ID Mask Node

The *ID Mask Node* can be used to access an alpha mask per object or per material.

See also

The ID Mask node is superseded by the [Cryptomatte Node](#). Cryptomatte is more feature complete and supported by Cycles and EEVEE. It is recommended to use this feature moving forward.



Inputs

ID Value

Input for the *Object Index* or *Material Index* render pass. Which is an output of the [Render Layers node](#) or the [Image node](#) with a multi-layer format.

Properties

Index

Selection of the previously specified index.

Anti-Aliasing

This post-processing filter smooths the mask edges. See [Anti-Aliasing](#).

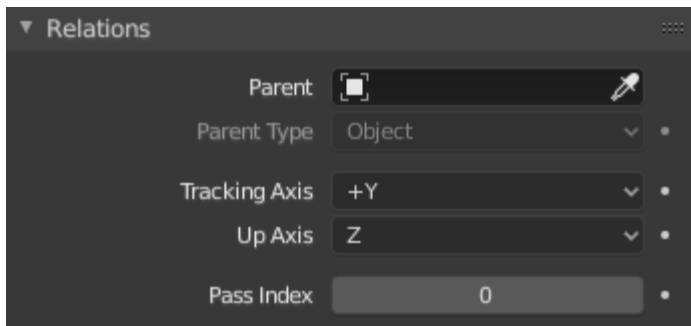
Outputs

Alpha

The mask is white where the object is and black where it is not. If the object is transparent, the alpha mask represent that with gray values.

Setup

An index can be specify for any object or material in the scene. The Object Index can be set in *Properties* ▶ *Object Properties* ▶ *Relations* ▶ *Pass Index* and *Material* ▶ *Settings* ▶ *Pass Index* for the Material Index. To be accessible after rendering, the Render Engine must be Cycles, and *Object Index* or *Material Index* render pass has to be enabled.



Object Pass Index.

Example

In this example, the left rear red cube is assigned Pass Index 1, and the right cube Pass Index 2. Where the two cubes intersect, there is going to be noticeable pixelation because they come together at a sharp angle and are different colors. Using the mask from object 1, which is smoothed (anti-aliased) at the edges, we use a *Mix Node* set on *Multiply* to multiply the smoothed edges of the image, thus removing those nasty lines, thus, being smoothed out.

Tracking Nodes

- [Plane Track Deform Node](#)
- [Stabilize 2D Node](#)
- [Track Position Node](#)

Plane Track Deform Node

The Plane Track Deform Node is used to replace flat planes in footage by another image, using plane tracks from motion tracking.



Plane Track

Before using this node, [plane track](#) for the footage should be made in the *Movie Clip Editor*.

Inputs

Image

Image to put in place of the plane track, and thus, override that area in the movie clip.

Properties

Movie Clip

Used to select the movie clip whose plane track to use. For controls see [Data-Block Menu](#).

Object

Used to select the object to which the plane track is linked.

Track

Used to select the plane track to use.

Motion Blur

Specify whether to use blur caused by motion of the plane track or not.

Samples

Set the number of samples to take for each frame. The higher this number, the smoother the blur effect, but the longer the render, as each virtual intermediate frame has to be rendered.

Note

Samples are taken only from the *next* frame, not the previous one. Therefore the blurred object will appear to be slightly ahead of how it would look without motion blur.

Shutter

Time (in frames) the shutter is open. If you are rendering at 24 fps, and the Shutter is set to 0.5, the time in between frames is 41.67 ms, so the shutter is open for half that, 20.83 ms.

Outputs

Image

The output by perspective wrapping the image to that plane track.

Plane

Produces a black-and-white mask of the plane track.

Examples

Using Image Output

This can simply be achieved by using the Alpha Over node.

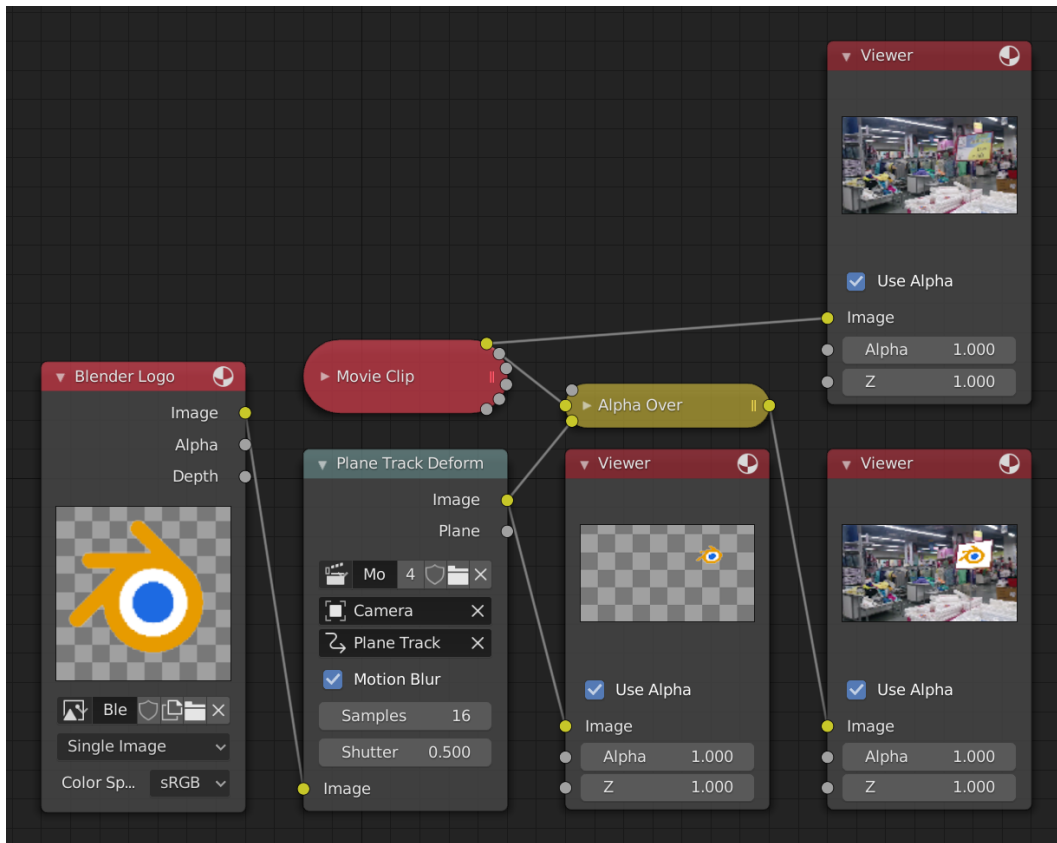
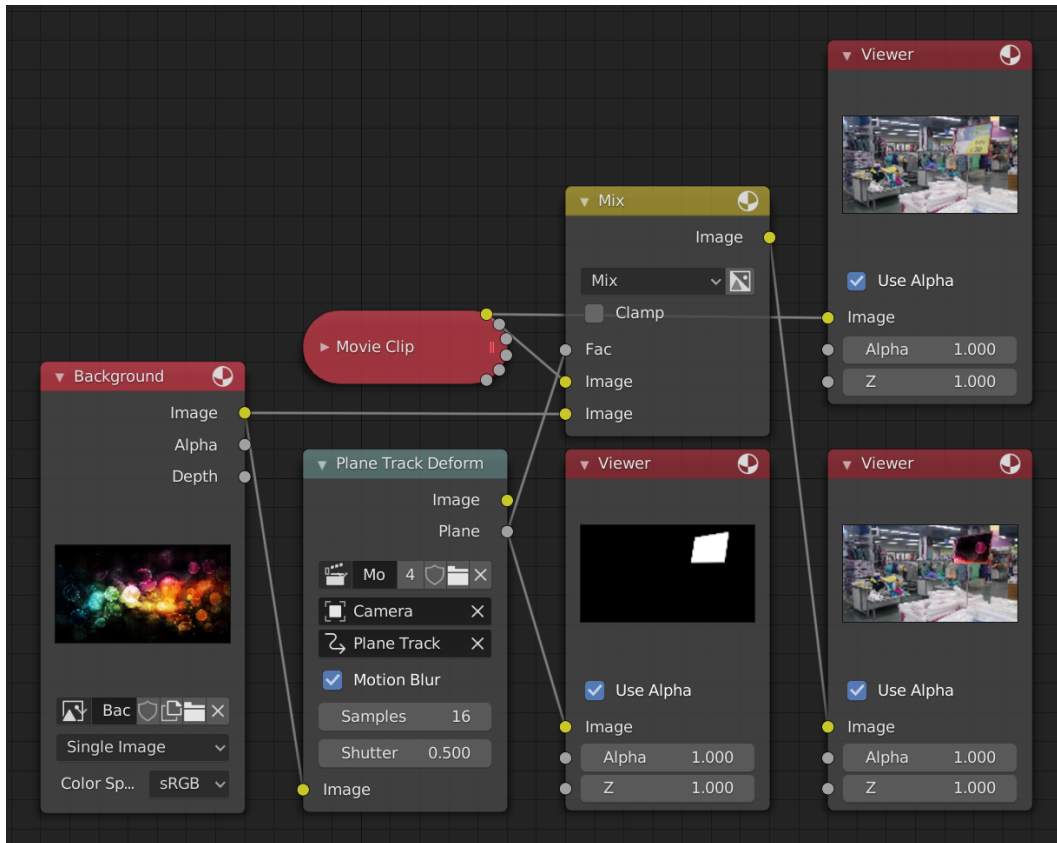


Image output.

Using Plane Output

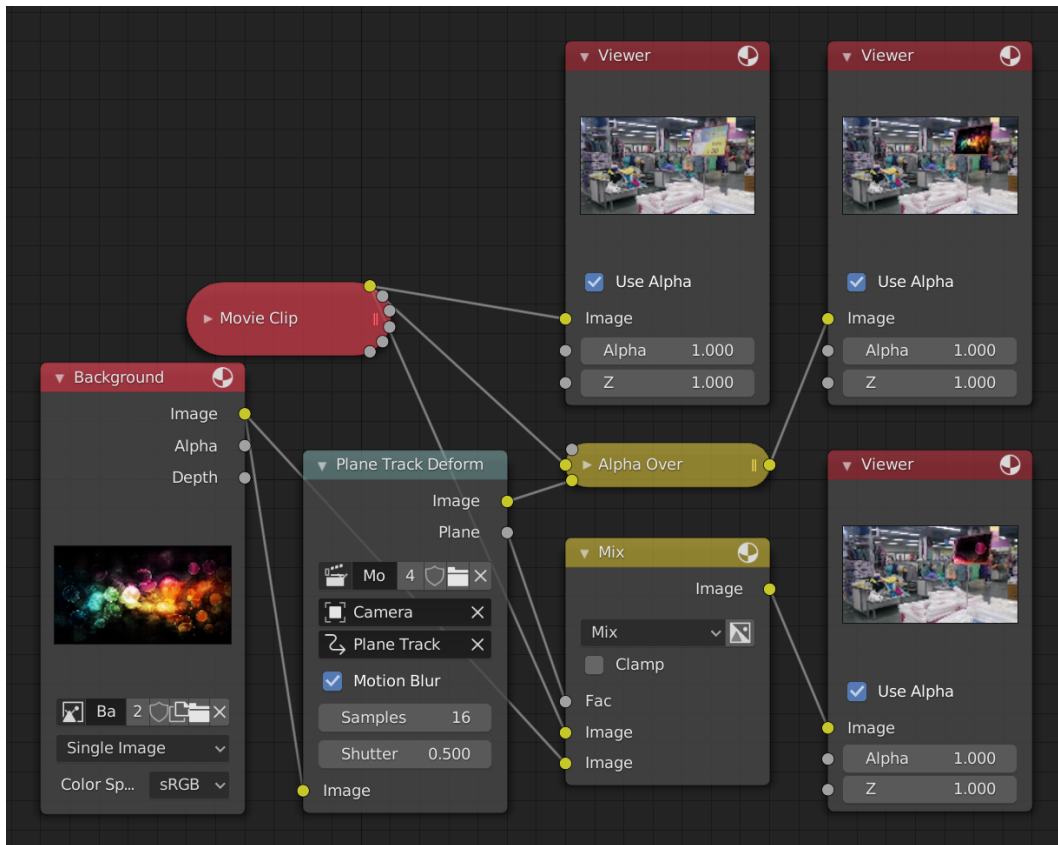
This can be achieved by mixing the movie clip and the image by using the plane output as the factor.



Plane output.

Using Image Output vs Using Original Image

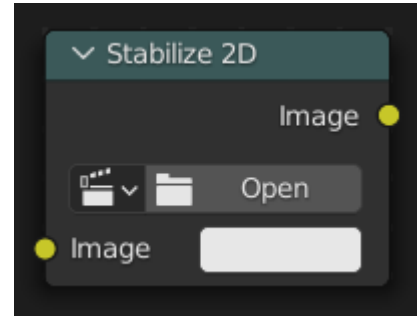
Using Image output scales, moves, and skews the input image according to the track while using the original image and mixing it with the movie clip using Plane output as factor will display the part of the image that lies inside that mask. This image shows the difference:



Comparison between image output and original image (see Viewer nodes carefully).

Stabilize 2D Node

Stabilizes the footage according to the settings set in *Movie Clip Editor* ▶ *Properties* ▶ *Stabilization* ▶ *2D Stabilization*. For more information, see [2D Stabilization](#).



Inputs

Image

Standard color input.

Properties

Movie Clip

The movie clip whose stabilization to use.

Interpolation

Various methods for the stabilization. Usually, the same as used in *Movie Clip Editor* ▶ *Properties* ▶ *Stabilization* ▶ *2D Stabilization* ▶ *Interpolate*. For technical details on their difference, [see this](#). But for most purposes, default of Bilinear should suffice.

Invert

Invert the stabilization. If the stabilization calculated is to move the movie clip up by 5 units, this will move the movie clip down by 5 units.

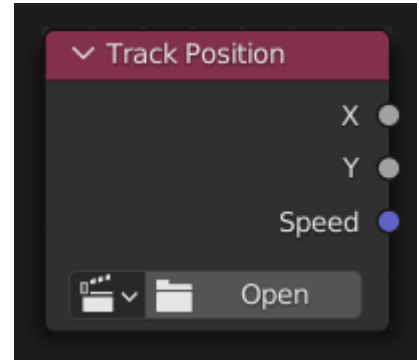
Outputs

Image

Standard color input.

Track Position Node

The *Track Position node* is used to return information about a tracking marker to the Compositor.



Inputs

This node has no inputs.

Properties

Movie Clip

Used to select a Movie Clip data-block to use, for controls see [Data-Block Menu](#).

Tracking Object

Camera object to get track information from.

Track Name

The name of the track to get track information from.

Position

Which marker position to use for output.

- | | |
|-------------------------|--|
| Absolute:: | Outputs an absolute position of a marker. |
| Relative Start:: | Outputs the positions of a marker relative to the first marker of a track. |
| Relative Frame:: | |

Outputs the positions of a marker relative to the markers of the given *Frame*.

Absolute Frame:: Outputs the absolute positions of a marker at the given *Frame*.

Outputs

X/Y

The marker's X and Y location.

Speed

The velocity of the marker, measured in pixels per frame. This could be used to fake effects like motion blur by connecting it to the Vector Blur Node.

Transform Nodes

These nodes distort the image in some fashion, operating either uniformly on the image, or by using a mask to vary the effect over the image.

- [Rotate Node](#)
 - [Scale Node](#)
 - [Transform Node](#)
 - [Translate Node](#)
-

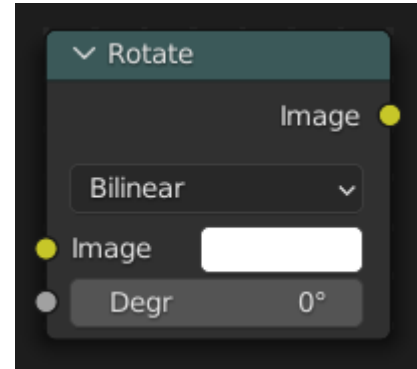
- [Corner Pin Node](#)
 - [Crop Node](#)
-

- [Displace Node](#)
 - [Flip Node](#)
 - [Map UV Node](#)
-

- [Lens Distortion Node](#)
- [Movie Distortion Node](#)

Rotate Node

This node rotates an image.



Inputs

Image

Standard color input.

Degr

Rotation angle in degree. Positive values rotate clockwise and negative ones counterclockwise.

Properties

Filter

Interpolation Methods.

- | | |
|-------------------|---|
| Nearest:: | No interpolation, uses nearest neighboring pixel. |
| Bilinear:: | Simple interpolation between adjacent pixels. |
| Bicubic:: | Highest quality interpolation. |

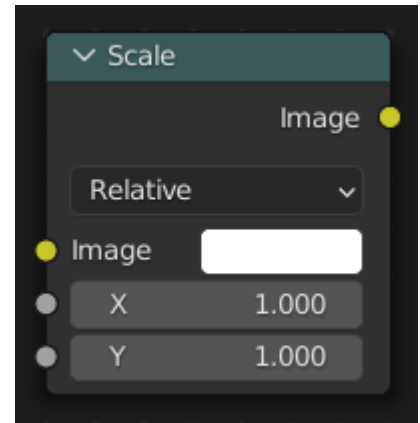
Outputs

Image

Standard color output.

Scale Node

This node scales the size of an image.



Inputs

Image

Standard color input.

X, Y

Scale in the axis directions, only available if *Space* is set to *Relative* or *Absolute*.

Properties

Space

Coordinate Space to scale relative to.

Relative

Percentage values relative to the dimensions of the image input.

Absolute

Size of an image by using absolute pixel values.

Scene Size

Sizes an image to the size of the final render resolution for the scene. For example, rendering a scene at the standard 1080p resolution but setting the render percentage at 50%, will produce a 1080p image with the scene scaled down 50% and leaving the rest of the image as alpha.

Render Size

Image dimensions set in the Render panel.

Stretch, Fit, Crop

Stretch distorts the image so that it fits into the render size. Fit scales the image until the bigger axis “fits” into the render size. Crop cuts the image so that it is the same aspect ratio as the render size.

X, Y

Offset factor for the final scaled image.

Outputs

Image

Standard color output.

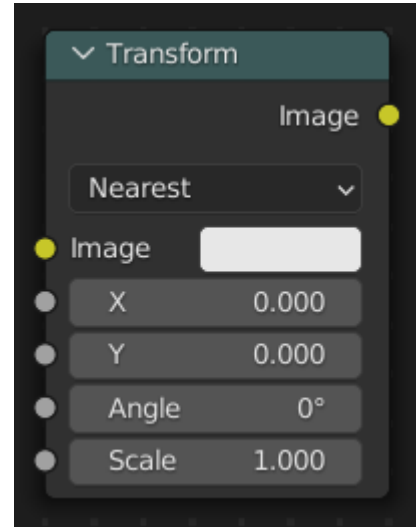
Examples

For instance X: 0.5 and Y: 0.5 would produce an image which width and height would be half of what they used to be.

Use this node to match image sizes. Most nodes produce an image that is the same size as the image input into their top image socket. To uniformly combine two images of different size, the second image has to be scaled up to match the resolution of the first.

Transform Node

This node combines the functionality of three other nodes: [Scale](#), [translate](#), and [rotate](#) nodes.



Inputs

Image

Standard color input.

X, Y

Used to move the input image horizontally and vertically.

Angle

Used to rotate an image around its center. Positive values rotate counter-clockwise and negative ones clockwise.

Scale

Used to resize the image. The scaling is relative, meaning a value of 0.5 gives half the size and a value of 2.0 gives twice the size of the original image.

Properties

Filter

Interpolation Methods.

Nearest::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

Outputs

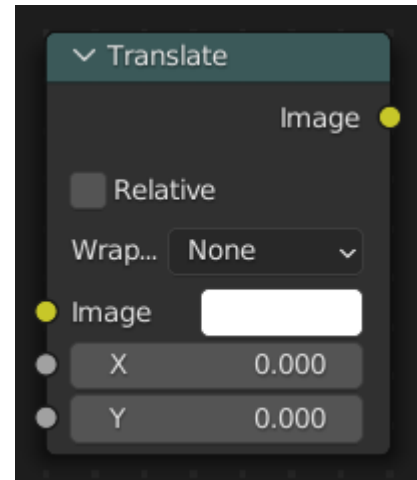
Image

Standard color output.

Translate Node

The Translate node moves an image.

Could also be used to add a 2D camera shake.



Inputs

Image

Standard color input.

X, Y

Used to move the input image horizontally and vertically.

Properties

Relative

Percentage translation values relative to the input image size.

Wrapping

Repeat pixels on the other side when they extend over the image dimensions, making endless translating possible.

None, X Axis, Y Axis, Both Axis

Filter

Interpolation Methods.

Nearest::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

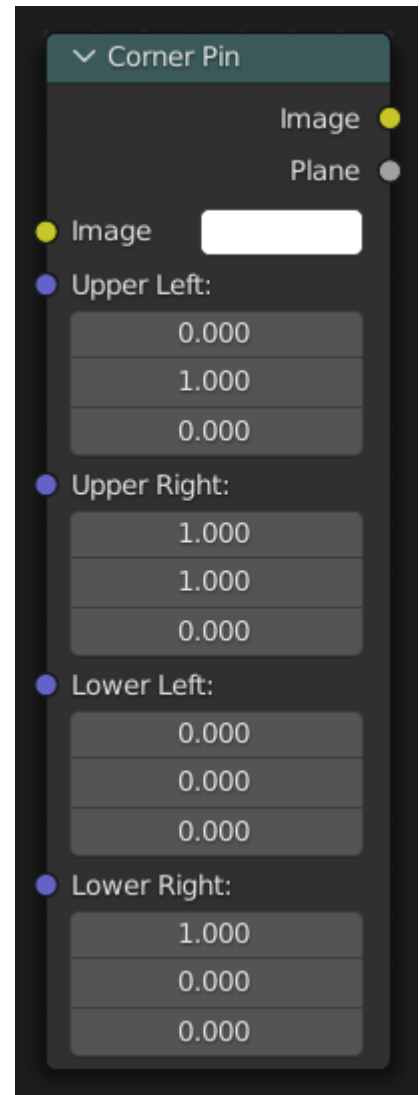
Outputs

Image

Standard color output.

Corner Pin Node

The Corner Pin node uses explicit corner values for a plane warp transformation. It works like the [Plane Track Deform](#) node, but without using “plane track” data from the Movie Clip Editor.



Inputs

Image
Standard color input.

Corners

Four vector inputs to define the plane warping. (Z component of vector inputs is ignored.)

Properties

This node has no properties.

Outputs

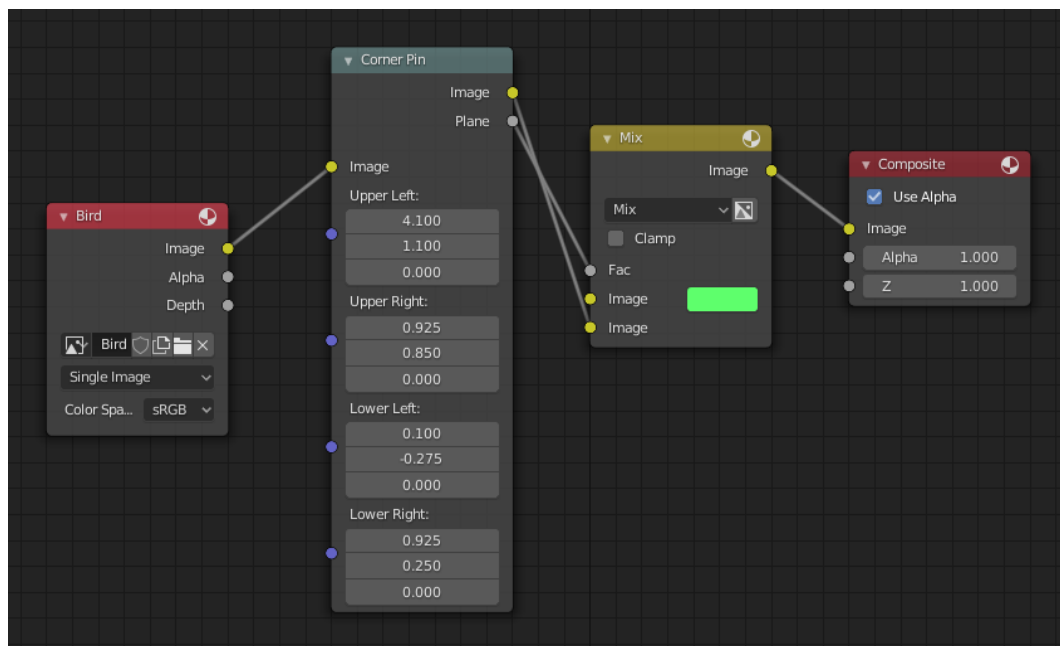
Image

Standard color output. (The image after distorting.)

Plane

A black-and-white alpha mask of the plane.

Example



An example of the Corner Pin node.



An example of the distorted image.

In the example above, the image of the bird is distorted by the vectors specified by the Corner Pin node.

Crop Node

The Crop node crops an input image to a selected region by either making the cropped area transparent or by resizing the input image.



Inputs

Image

Standard color input. If no image is selected, an image filled with the selected color is used. You can use and crop this image in combination with another background image.

Properties

Crop Image Size

When disabled, the image remains the same size, but the cropped areas become transparent pixels. When enabled, the image size is cropped to the specified region and gets a new width or height or both.

Note that this will probably reposition the image in the render output because the cropped image is automatically centered.

Relative

When enabled, crop dimensions are a percentage of the input image's width and height. When disabled, the range of the *Crop Region Values* are the width and height of the image in pixels.

Crop Region Values

Define borders of the crop region; *Left* or *Right* can vary between 0 and the width of the image. *Up* or *Down* can vary between 0 and the height of the image.

Note

The terminology (*Left*, *Right*, *Up*, *Down*) can be misunderstood easily. First, the numbers do not represent the amount of cropping, e.g. *Left* is set to 50 and *Right* to 50 does not mean that you will be cropping the image for 50 pixels on both the left and right side. In fact, it will result in zero-sized image because you are cropping from pixel 50 to pixel 50. So, the numbers defines a position in the input image.

Secondly, depending on which one is bigger, *Left* should be interpreted as *Right* and vice versa. If *Left* is greater than *Right* then both values are switched and *Left* gets the value of *Right* and vice versa. The same operation is done for *Up* and *Down*, where you can think of them as the top and bottom of the image.

Thirdly, the terms *Up* and *Down* are ambiguous and suggest an action; e.g. "Crop down". The values, however, are not amounts but positions. The term *Down* should be interpreted as "Bottom" and *Up* as "Top".

Outputs

Image

Standard color output.

Usage

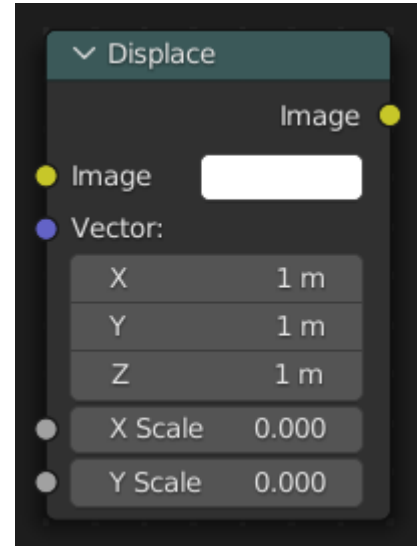
The following workflow removes some possible confusion:

1. Uncheck *Crop Image Size* for this step, so that you can see the borders of the input image. To see this border, you have to select the Viewer node.
2. If you don't need pixel-perfect cropping, check *Relative* so that you do not have to consider the exact dimensions of the input image.
3. Set *Left* and *Down* to zero. Set *Right* and *Up* to one, or to the width and height of the input image. You should see now the entire input image in the backdrop. *Up* is thus interpreted as the top of the image. The origin of the image (0, 0) is at the bottom (down) left corner.
4. To crop from the left, change the *Left* value. To crop from the right, change the *Right* value. To crop from the top, change the *Up* value. To crop from the bottom, change the *Down* value.

Displace Node

The *Displace Node* displaces the pixel position based on an input vector.

This node could be used to model phenomena, like hot air distortion, refractions of uneven glass or for surreal video effects.



Inputs

Image

Standard color input.

Vector

Input of the displacement map. If the a color output is implicitly converted in the vector input, the first channel (red) value determines displacement along X axis. The second channel (green) the displacement along Y axis. If the input is a grayscale image, where both the channel values are equal, the input image will be displaced equally in both X and Y directions.

Scale X, Y

Separate scaling of the vector input in X and Y direction. Acting as multipliers by increasing or decreasing the strength of the displacement along their respective axes.

Properties

This node has no properties.

Outputs

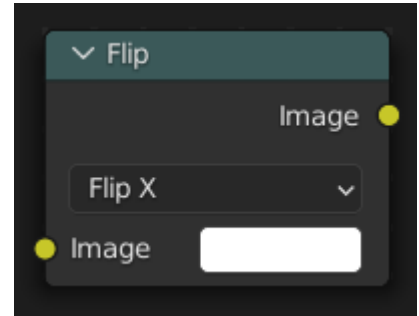
Image

Standard color output.

Flip Node

This node flips an image along a defined axis.

You can use this node to just flip or use it as a part of mirror setting. Mix half of the image to be mirrored with its flipped version to produce mirrored image.



Inputs

Image

Standard color input.

Properties

Axis

This can be either X or Y. Also, flipping can be done on both X and Y axis simultaneously.

Flip X, Flip Y, Flip X & Y

Outputs

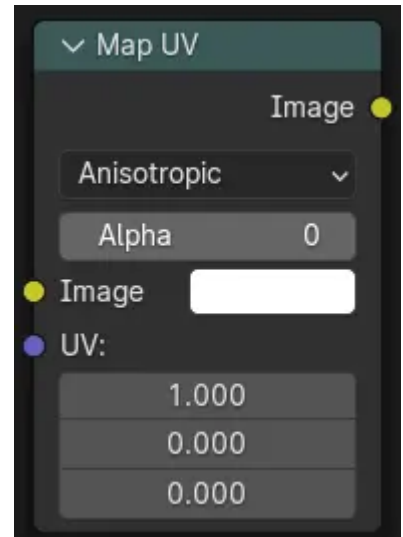
Image

Standard color output.

Map UV Node

Map a texture using UV coordinates, to apply a texture to objects in compositing.

May be used in combination with [Cryptomatte Node](#) to apply the texture only to specific objects.



Inputs

Image

The new 2D texture.

UV

The input for UV render pass. See [Cycles render passes](#).

Hint

To store the UV pass a multi-layer OpenEXR format could be used.

Properties

Filter Type

Interpolation Methods.

Anisotropic:: Enhances the clarity of textures viewed at oblique angles, addressing issues like blurring and distortion.

Nearest:: No interpolation, uses nearest neighboring pixel.

Alpha

Alpha threshold is used to fade out pixels on boundaries.

Outputs

Image

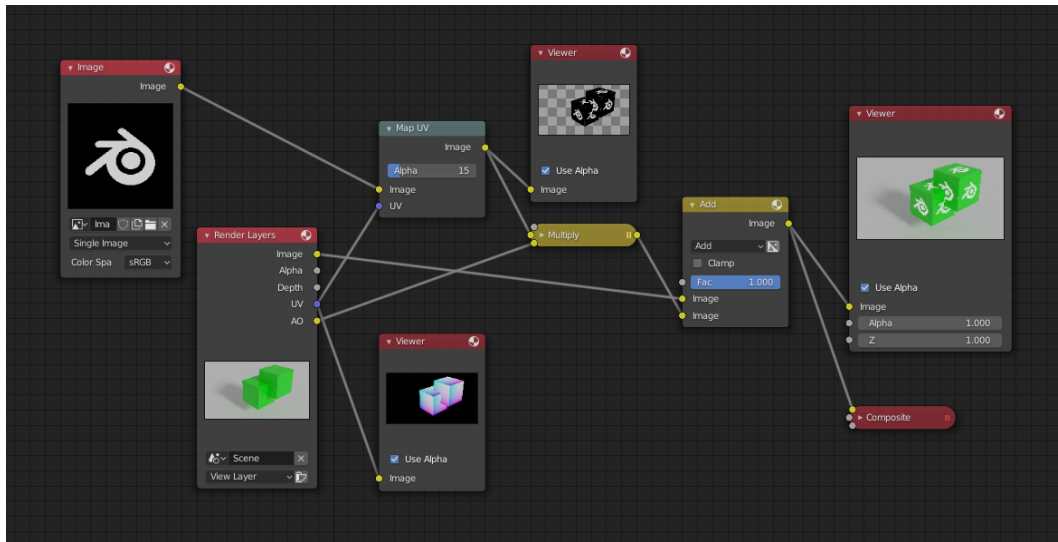
The resulting image is the input image texture distorted to match the UV coordinates. That image can then be overlay mixed with the original image to paint the texture on top of the original. Adjust alpha and the mix factor to control how much the new texture overlays the old.

Hint

When painting the new texture, it helps to have the UV maps for the original objects in the scene, it is recommended to keep those UV texture outlines around even, when shooting is done.

Examples

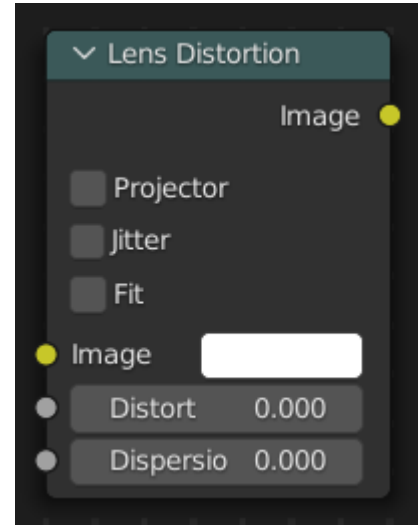
In the example below, we have overlaid a grid pattern on top of the two heads after they have been rendered. During rendering, we enabled the UV layer in the Properties *Render Layer* ▶ *Passes*. Using a Mix node (“Overlay” in figure), we mix that new UV texture over the original face. We can use this grid texture to help in any motion tracking that we need to do.



Adding UV textures in post-production.

Lens Distortion Node

Use this node to simulate distortions that real camera lenses produce.



Inputs

Image

Standard color input.

Distortion

This creates a bulging or pinching effect from the center of the image.

Dispersion

This simulates chromatic aberrations, where different wavelengths of light refract slightly differently, creating a rainbow colored fringe.

Properties

Projector

Enable or disable slider projection mode. When on, distortion is only applied horizontally. Disables *Jitter* and *Fit*.

Jitter

Adds jitter to the distortion. Faster, but noisier.

Fit

Scales image so black areas are not visible. Only works for positive distortion.

Outputs

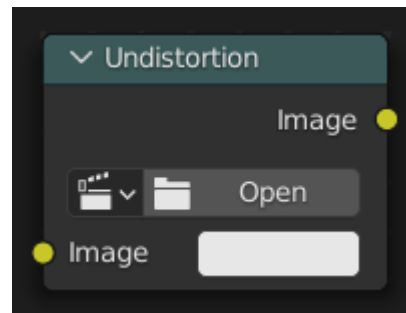
Image

Standard color output.

Movie Distortion Node

In the real world, all camera lenses produce some or the other sort of lens distortion. But, whatever we render has got no distortion. So, this node helps in removing distortion from movies or adding distortion to render to make our render blend in with the movie clip.

Usually, it is used while motion tracking.



Calculating Distortion

Before using this node, one has to calculate the lens distortion of the clip. This can be done by adjusting K1, K2 and K3 values in *Movie Clip Editor* ▶ *Properties* ▶ *Lens*. For more information on how to edit those values, [check this out](#).

Inputs

Image

Standard color input.

Properties

Movie Clip

Used to select the movie clip whose distortion is to be used. This can be useful if more than one movie clips are present, each having a different distortion setting. For controls see [Data-Block Menu](#).

Distortion Method

Undistort::

Used to undistort the image received, and is usually used for the raw distorted movie clip.

Distort::

Used to distort the image received, and is usually used for rendered images.

Outputs

Image

The image after distorting/undistorting.

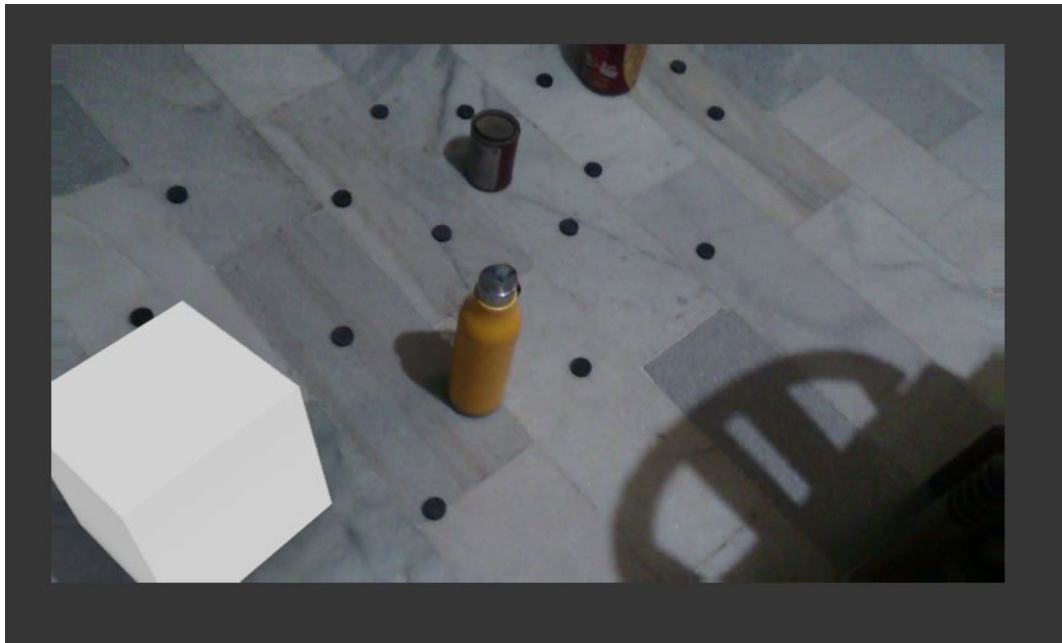
Distortion vs Undistortion

Although, both, distortion of render and undistortion of movie clip are possible, and produce similar results, there is a difference between these two methods.

There are two kinds of lens distortion possible and, in simple terms, they can be said as:

1. When the movie clip is bulging out.
2. When the movie clip is bulging in.

For the first case, it is recommended to distort the render and leave the movie clip as it is, because, undistorting the movie clip will require extra pixel information, which is not available to Blender. Similarly, in the second case, it is recommended to undistort the movie clip and leave the render as it is, because, distorting the render will require those extra unavailable pixels. Doing the wrong method in the wrong case can create weird results around the edges, such as in the image shown.



Problems (notice the edges?)

Utilities Nodes

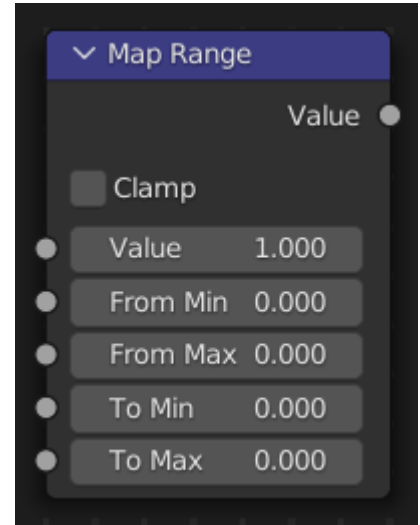
- [Map Range Node](#)
- [Map Value Node](#)
- [Math Node](#)

-
- [Levels Node](#)
 - [Normalize Node](#)

-
- [Split Node](#)
 - [Switch Node](#)
 - [Switch View Node](#)

Map Range Node

This node converts (maps) an input value range into a destination range. By default, values outside the specified input range will be proportionally mapped as well. This node is similar to *Map Value* node but provides a more intuitive way to specify the desired output range.



Inputs

Value

The input value to be remapped.

From Min

The lower bound of the range to remap from.

From Max

The higher bound of the range to remap from.

To Min

The lower bound of the target range.

To Max

The higher bound of the target range.

Properties

Clamp

Clamps values to Min/Max of the destination range.

Outputs

Value

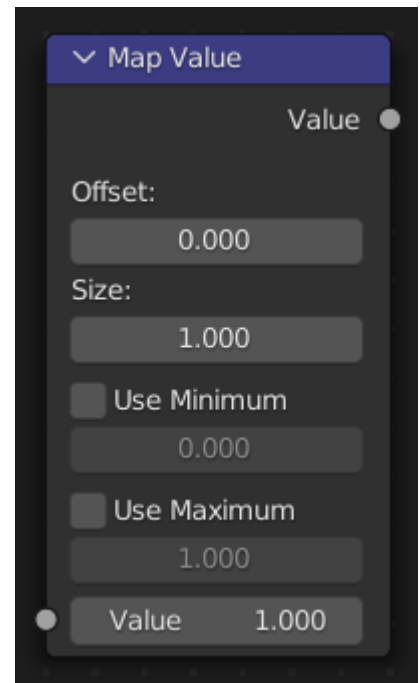
Standard value output.

Usage

One important use case is to easily map the original range of the Z-depth channel to a more usable range (i.e: 0.0 - 1.0) for use as a matte for colorization or filtering operations.

Map Value Node

Map Value node is used to scale, offset and clamp values.



Inputs

Value

Standard Value input. (Value refers to each vector in the set.)

Properties

Offset

Factor added to the input value.

Size

Scales (multiply) the input value.

Use Minimum, Maximum

Enable this to activate their related operation.

Min, Max

Defines a range between minimum and maximum to [Clamp](#) the input value to.

Outputs

Value

Standard value output.

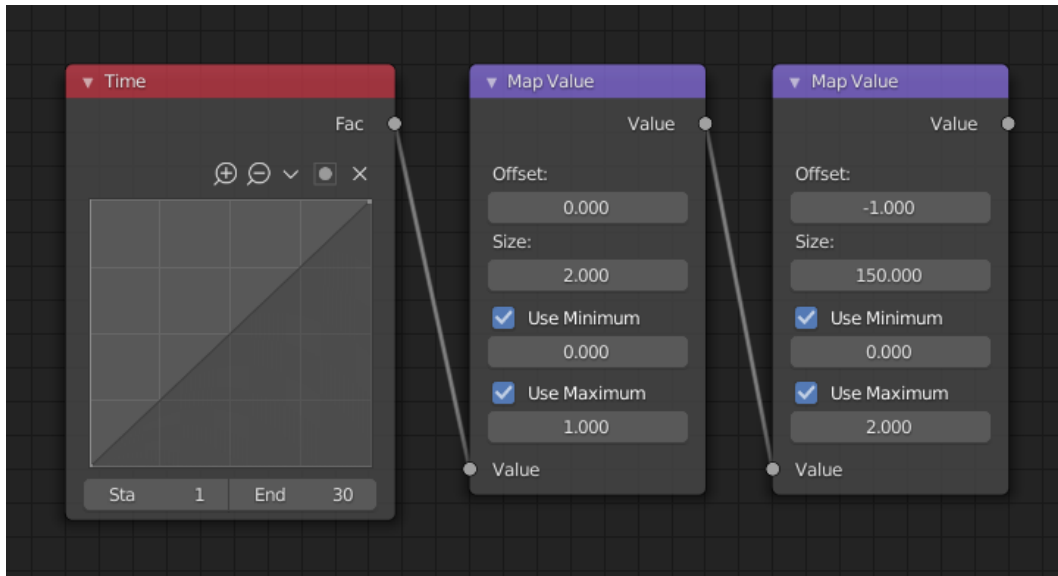
Example

Z-Depth Map

This is particularly useful in achieving a depth of field effect, where the Map Value node is used to map a Z value (which can be 20 or 30 or even 500 depending on the scene) to the range between (0 to 1), suitable for connecting to a Blur node.

Multiplying Values

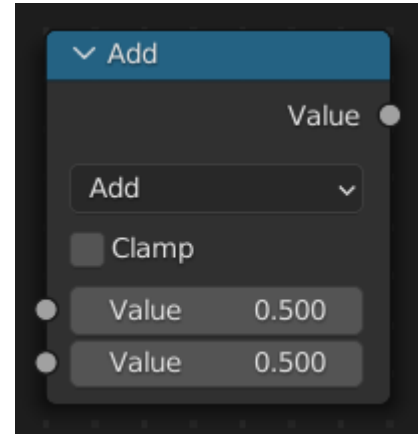
The Map Value node can also be used to multiply values to achieve a desired output value. In the mini-map to the right, the Time node outputs a value between 0.0 and 1.0 evenly scaled over 30 frames. The *first* Map Value node multiplies the input by 2, resulting in an output value that scales from 0.0 to 2.0 over 30 frames. The *second* Map Value node subtracts 1 from the input, giving working values between (-1.00 to 1.0), and multiplies that by 150, resulting in an output value between (-150 to 150) over a 30-frame sequence.



Using Map Value to multiply.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value

Input Value. Trigonometric functions read this value as radians.

Addend

Input Addend.

Base

Input Base.

Exponent

Input Exponent.

Epsilon

Input Epsilon.

Distance

Input Distance.

Min

Input Minimum.

Max

Input Maximum.

Increment

Input Increment.

Scale

Input Scale.

Degrees

Input Degrees.

Radians

Input Radians.

Properties

Operation

The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with <i>Addend</i> .
Power::	The <i>Base</i> raised to the power of <i>Exponent</i> .
Logarithm::	The log of the value with a <i>Base</i> as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

Absolute:: One divided by the square root of the value.
The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises [Euler's number](#) to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: [Smooth Minimum](#).

Smooth Maximum:: [Smooth Maximum](#).

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo:: Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

	Outputs a value between <i>Min</i> and <i>Max</i> based on the absolute difference between the input value and the nearest integer multiple of <i>Max</i> less than the value.
Snap::	Rounds the input value down to the nearest integer multiple of <i>Increment</i> .
Ping-pong::	Bounces back and forth between 0.0 and the <i>Scale</i> as the input value increases.
Trigonometric	
Sine::	The Sine of the input value.
Cosine::	The Cosine of the input value.
Tangent::	The Tangent of the input value.
Arcsine::	The Arcsine of the input value.
Arccosine::	The Arccosine of the input value.
Arctangent::	The Arctangent of the input value.
Arctan2::	Outputs the Inverse Tangent of the first value divided by the second value measured in radians.
Hyperbolic Sine::	The Hyperbolic Sine of the input value.
Hyperbolic Cosine::	The Hyperbolic Cosine of the input value.
Hyperbolic Tangent::	The Hyperbolic Tangent of the input value.
Conversion	
To Radians::	Converts the input from degrees to radians.
To Degrees::	Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See [Clamp](#).

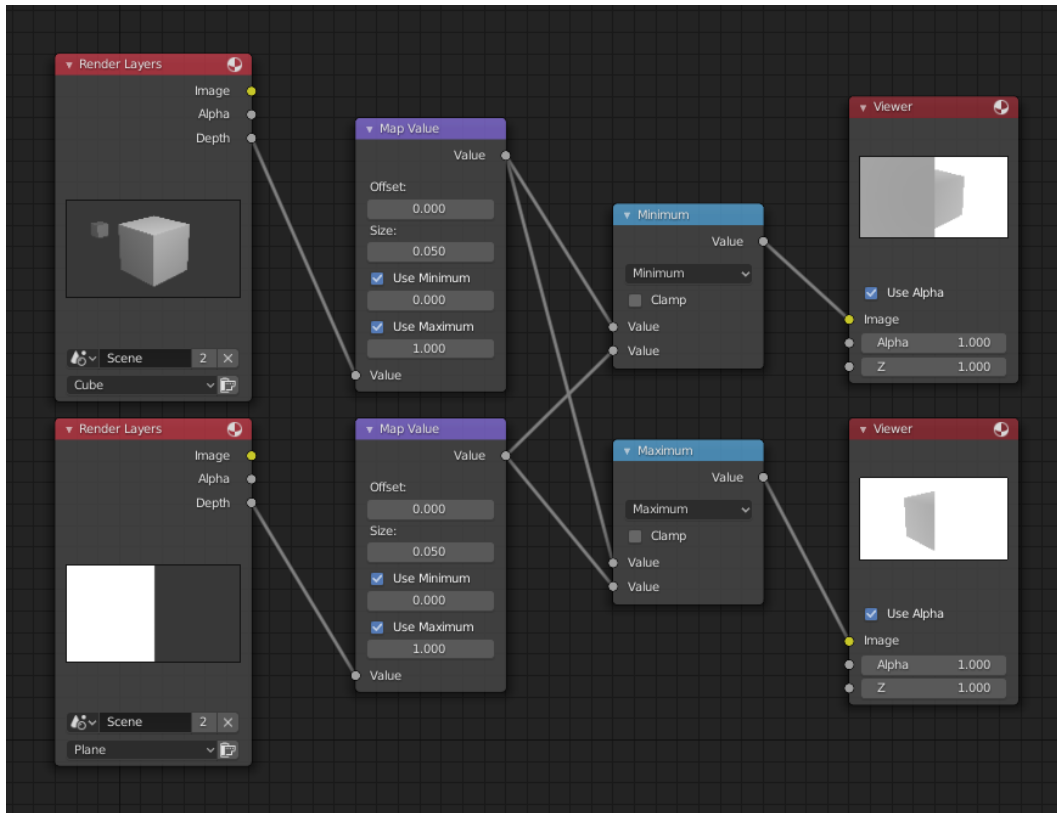
Outputs

Value

Numerical value output.

Examples

Manual Z-Mask



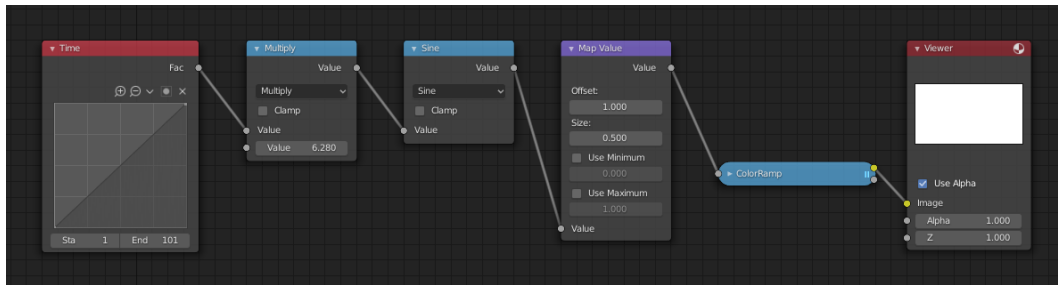
Minimum and maximum function example.

The top *Render Layers* node has a cube that is about 10 units from the camera. The bottom *Render Layers* node has a plane that covers the left half of the view and is 7 units from the camera. Both are fed through their respective *Map Value* nodes to multiply the depth value by 0.05 and clamp it to $[0.0, 1.0]$, bringing it into a suitable range for displaying it as a color.

The *Minimum* node selects the smallest of the two depth values at each pixel. In the left half, it chooses the plane (because it's closer than the cube), and in the right half, it chooses the cube (because it's closer than the background, which is infinitely far away).

The *Maximum* node selects the largest of the two depth values at each pixel. In the left half, it chooses the cube (because it's farther away than the plane), and in the right half, it chooses the background (because it's farther away than the cube).

Using Sine Function to Pulsate



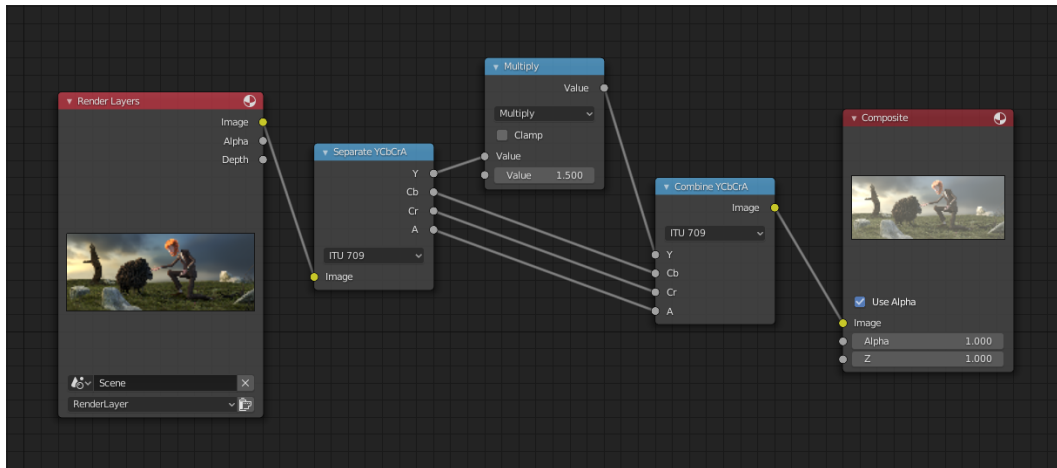
Using sine function example.

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. At frame 25, the output value is 0.25. That value is multiplied by $2 \times \pi$ (6.28) and converted to 1.0 by the Sine function, since $\sin(2 \times \pi / 4) = \sin(\pi / 2) = +1.0$.

Since the sine function can output values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by 0.5 (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a gray-scale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, $\sin(\pi / 2) = 1.0$. Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color “pulse”.

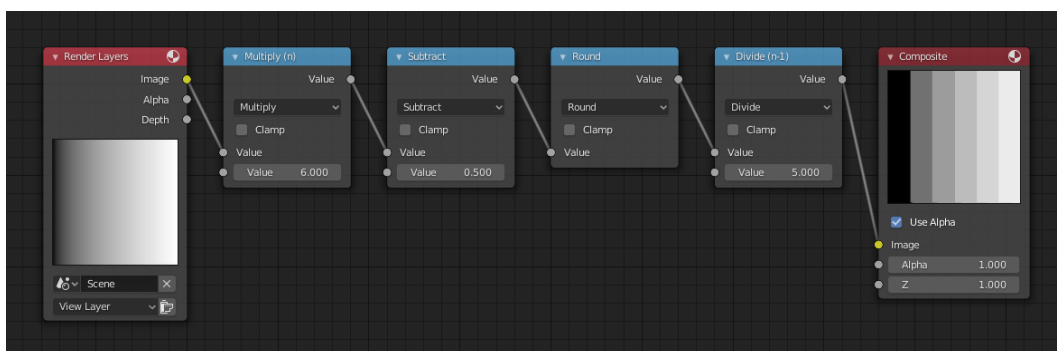
Brightening (Scaling) a Channel



Scaling a channel example.

This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value* node with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high dynamic range image. For this particular example, there is also a *Brightness/Contrast* node that might give simpler control over brightness.

Restrict Color Selection (Posterization)



Posterization example.

In this example, we restrict the color values to be one of the six values: 0, 0.2, 0.4, 0.6, 0.8, 1.

To split up a continuous range of values between 0 and 1 to certain set of values, the following function is used: $\lfloor \text{round}(x \times n - 0.5) / (n - 1) \rfloor$, where

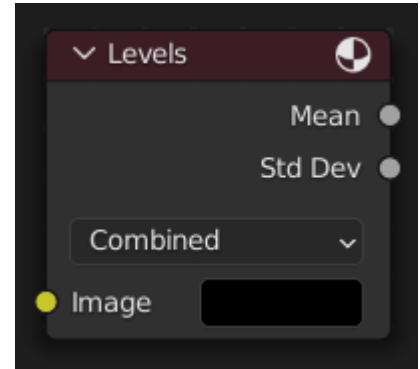
“n” is the number of possible output values, and “x” is the input pixel color.
[Read more about this function.](#)

To implement this function in Blender, consider the node setup above. We string the Math nodes into a function that takes each color (values from 0 to 1), multiplies it up by six, the desired number of divisions (values become from 0 to 6), offsets it by 0.5 (-0.5 to 5.5), rounds the value to the nearest whole number (produces 0, 1, 2, 3, 4, 5), and then divides the image pixel color by five (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

In the case of a color image, you need to split it into separate RGB channels using *Separate/Combine Color* nodes and perform this operation on each channel independently.

Levels Node

The Levels Node read the input color channels and outputs analytical values. The output is one-dimensional meaning the visualization will be a uniform gray color.



Inputs

Image
Standard color input.

Properties

Channel
Selects which color values are used to calculate analytics.

Combined::	Calculate values based on the red, green, and blue channels.
Red::	Calculate values based on the red channel.
Green::	Calculate values based on the green channel.
Blue::	Calculate values based on the blue channel.
Luminance::	Calculate values based on the Luminance of the image.

Outputs

Mean

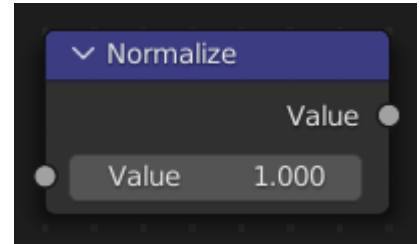
The mean is the average value of all image pixels in specified channel. It represents the overall brightness of the image and can be used as such for setups that depend on how “bright” or “dark” the input is.

Standard Deviation

How much pixel values differ from the mean. A low standard deviation indicates that the pixel values tend to be very close to the mean. A high standard deviation indicates that the values are spread out over a large range of values.

Normalize Node

Find the minimum and maximum values of a single channel. Then map the values to a range of 0 and 1. This is mostly useful for the Z-buffer.



Inputs

Value

Standard value input.

Properties

This node has no properties.

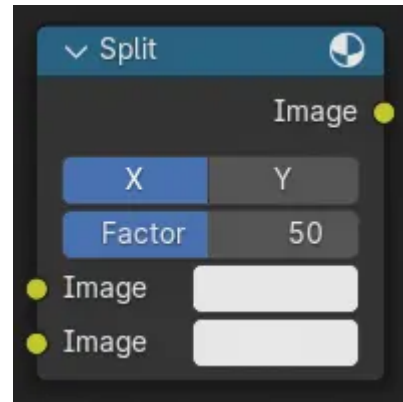
Outputs

Value

Standard value output.

Split Node

The *Split* node combines two images for side-by-side display. Typically used in combination with the [Viewer Node](#).



Inputs

Image

Shown on the right or top half set by the axis.

Image

And respectively the left or bottom half.

Properties

Axis

X or Y used as the split axis.

Factor

Percentage factor setting the space distribution between the two images.

Outputs

This node has no output sockets.

Hint

This node could be used to plan scene transitions by comparing the end frame of one scene with the start frame of another to make sure they align.

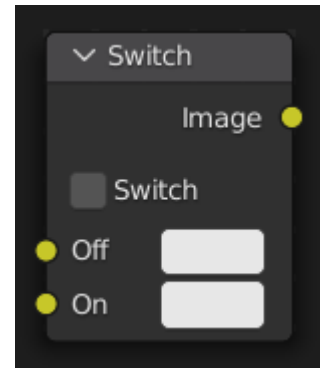
Examples



Example of a Split Viewer node.

Switch Node

Switch between two images using a checkbox.



Inputs

Image

First image input.

Image

Second image input.

Properties

Switch

- When it is unchecked, the first input labeled “Off” is passed to the output.
- When checked, the second input labeled “On” is passed to the output.

Outputs

Image

Standard color output.

Tip

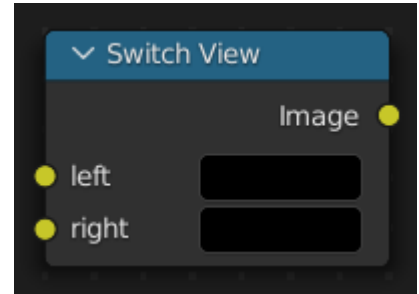
Switch state may be animated by adding a [keyframe](#). This makes the Switch node useful for bypassing nodes which are not wanted during part of a sequence.

Switch View Node

The *Switch View* node combines the *views* (left and right) into a single stereo 3D output. This can be useful if for example, you need to treat the view as separate images by combining each of the views.

See also

[The multi-view workflow.](#)



Inputs

Left

Left-eye image input.

Right

Right-eye image input.

Properties

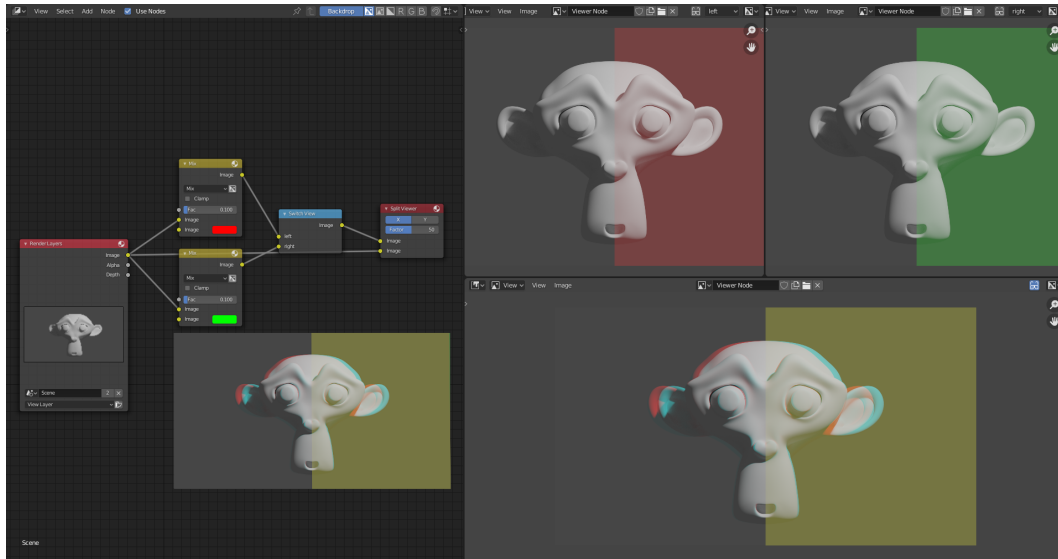
This node has no properties.

Outputs

Image

Stereo 3D image output.

Example



Compositor, Backdrop and Split Viewer Node.

The views to render are defined in the current scene views, in a similar way as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Vector Nodes

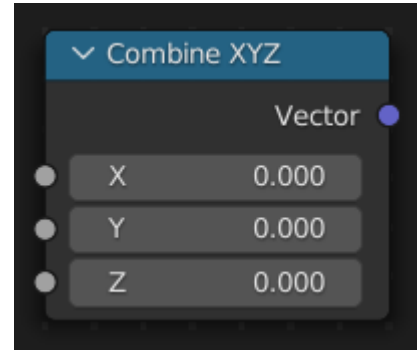
These nodes can be used to manipulate various types of vectors, such as surface normals and speed vectors.

- [Combine XYZ Node](#)
- [Separate XYZ Node](#)

-
- [Normal Node](#)
 - [Vector Curves Node](#)

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.



Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

Vector

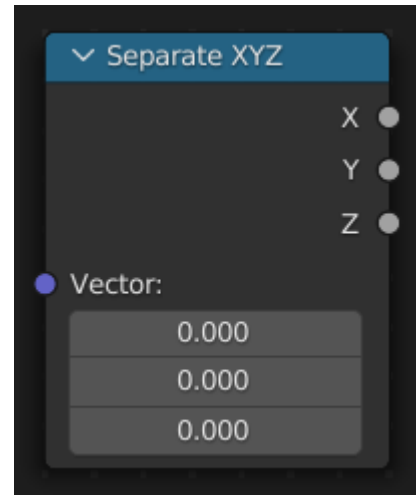
Standard vector output.

Note

The vector is not normalized.

Separate XYZ Node

The *Separate XYZ Node* splits a vector into its individual components.



Input

Vector

Standard vector input.

Properties

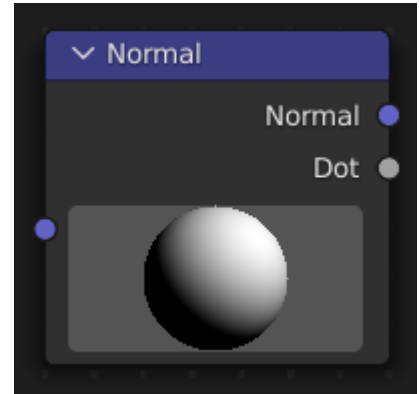
This node has no properties.

Outputs

- X
- Y
- Z

Normal Node

The Normal node generates a normal vector and a dot product.



Inputs

Normal
Normal vector input.

Properties

Normal Direction
To manually set a fixed normal direction vector. LMB click and drag on the sphere to set the direction of the normal. Holding **Ctrl** while dragging snaps to 45 degree rotation increments.

Outputs

Normal
Normal vector output.

Dot
Dot product output. The dot product is a scalar value.

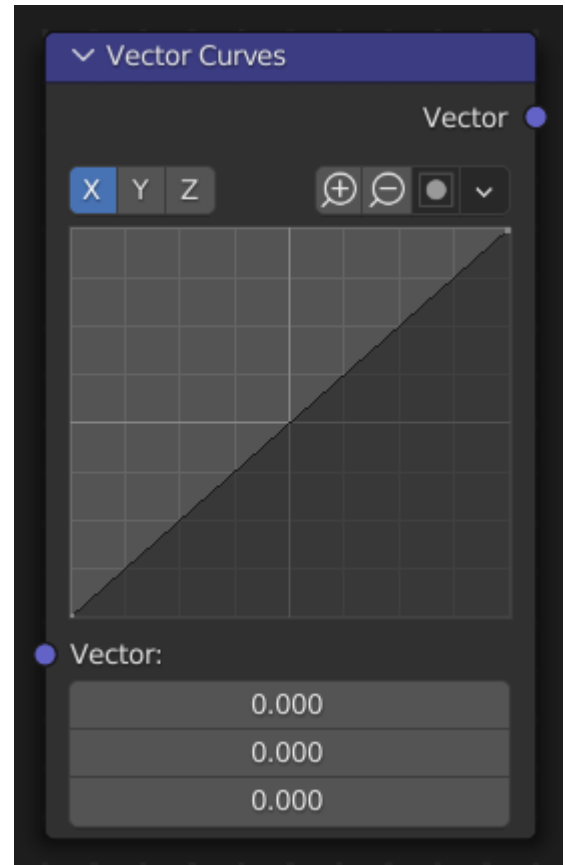
- If two normals are pointing in the same direction the dot product is 1.

- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.



Inputs

In the shader context the node also has an additional Factor property.

Factor

Controls the amount of influence the node exerts on the output vector.

Vector

Standard vector input.

Properties

Channel

X, Y, Z

Curve

For the curve controls see: [Curve widget](#).

Outputs

Vector

Standard vector output.

Group

A [Group Node](#) combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

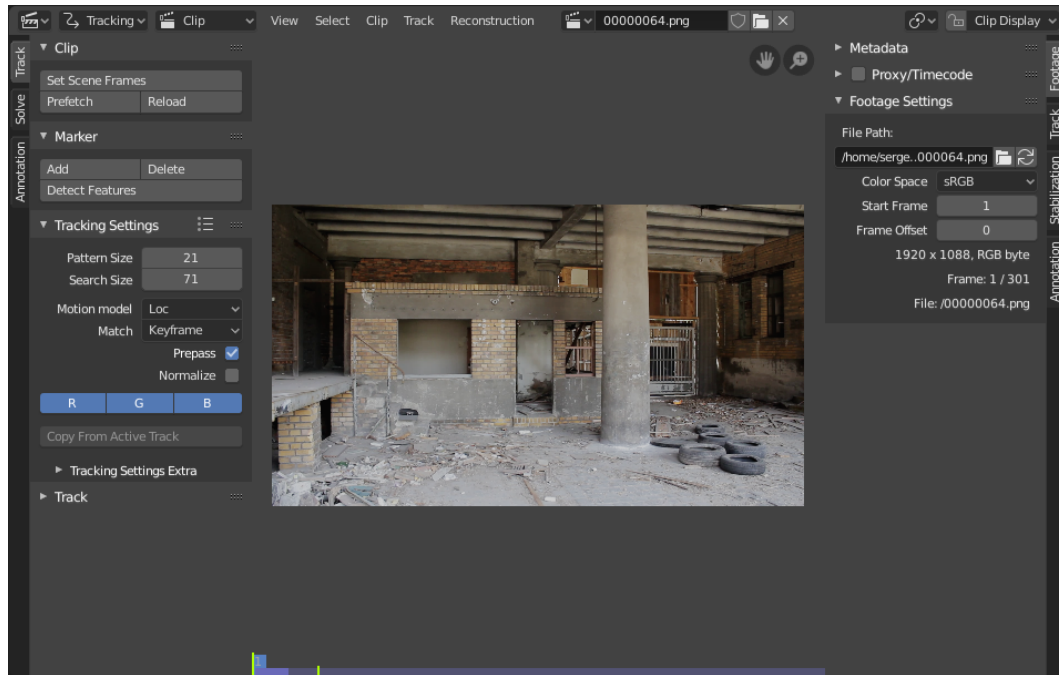
This section lists all the node groups, both those in the current blend-file and those [Linked or Appended](#) from another blend-file.

Layout Nodes

These are nodes which help you control the layout and connectivity of nodes within the Compositor.

Motion Tracking & Masking

You perform masking and tracking with the Movie Clip Editor.



The Movie Clip Editor.

See [Movie Clip Editor](#) for more information on the Movie Clip Editor.

- [Motion Tracking](#)
 - [Introduction](#)
 - [Clip View](#)
 - [Graph View](#)
 - [Dope Sheet View](#)
- [Masking](#)
 - [Introduction](#)
 - [S-Curves](#)
 - [Selecting](#)
 - [Editing](#)
 - [Sidebar](#)

Motion Tracking

- [Introduction](#)
 - [Views](#)
 - [Manual Lens Calibration](#)
 - [Camera and Object Motion Solving](#)
 - [Tools for Scene Orientation and Stabilization](#)
- [Clip View](#)
 - [Introduction](#)
 - [Tracking Marker](#)
 - [Toolbar](#)
 - [Selecting](#)
 - [Editing](#)
 - [Sidebar](#)
- [Graph View](#)
 - [Introduction](#)
 - [Header](#)
 - [Usage](#)
- [Dope Sheet View](#)
 - [Header](#)
 - [Usage](#)

Introduction

Motion Tracking is used to track the motion of objects and/or a camera and, through the constraints, to apply this tracking data to 3D objects (or just one), which have either been created in Blender or imported into the application. Blender's motion tracker supports a couple of very powerful tools for 2D tracking and 3D motion reconstruction, including camera tracking and object tracking, as well as some special features like the plane track for compositing. Tracks can also be used to move and deform masks for rotoscoping in the Mask Editor, which is available as a special mode in the Movie Clip Editor.

Views

In Tracking Mode there are three different views available. You can toggle between view modes using the View selector, which is located in the header. When you selected a view in the whole area of the Movie Clip editor will change. Hence, to display a curve or dope sheet view, the editor must be split into two, with one switched to the curve or dope sheet view.

Manual Lens Calibration

All cameras record distorted video. Nothing can be done about this because of the manner in which optical lenses work. For accurate camera motion, the exact value of the focal length and the “strength” of distortion are needed.

Currently, focal length can be automatically obtained only from the camera's settings or from the EXIF information. There are some external tools which can help to find approximate values to compensate for distortion. There are also fully manual tools where you can use a grid which is getting affected by distortion model and deformed cells defines straight lines in the footage.

Within Blender you can use the Annotation tool for this – just draw a line which should be straight on the footage using poly line brush and adjust the distortion values to make the annotations match lines on the footage.

To calibrate your camera more accurately, use the Grid calibration tool from OpenCV. OpenCV is using the same distortion model, so it should not be a problem.

Camera and Object Motion Solving

Blender not only supports the solving of camera motion, including tripod shots, but also the solving of object motion in relation to the motion of the camera. In addition to that there is the Plane Track, which solves the motion of all markers on one plane.

Tools for Scene Orientation and Stabilization

After solve, you need to orient the real scene in the 3D scene for more convenient compositing. There are tools to define the floor, the scene origin, and the X/Y axes to perform scene orientation.

Sometimes, the video footage includes spurious jumps and tilting movements, like e.g. when using a hand-held camera. Based on some tracked image elements, the [2D Stabilization](#) is able to detect and compensate such movements to improve the quality of the final result.

Clip View

- [Introduction](#)
 - [Main View](#)
- [Tracking Marker](#)
 - [Point](#)
 - [Plane](#)
- [Toolbar](#)
 - [Track](#)
 - [Solve](#)
- [Selecting](#)
 - [Box Select](#)
 - [Circle Select](#)
 - [Lasso Select](#)
 - [Select Grouped](#)
 - [Select Stabilization Tracks](#)
 - [Select Stabilization Rotation Tracks](#)
- [Editing](#)
 - [Clip](#)
 - [Track](#)
 - [Reconstruction](#)
- [Sidebar](#)
 - [Track](#)
 - [2D Stabilization](#)
 - [View](#)

Introduction

The Clip View is the main part of the Movie Clip editor; almost all motion tracking tools are concentrated within the Clip View.

It should be mentioned that the camera solver consists of three quite separate steps:

1. 2D tracking of footage.
2. Camera intrinsics (focal length, distortion coefficients) specification/estimation/calibration.
3. Solving camera, scene orientation, and scene reconstruction.

Main View

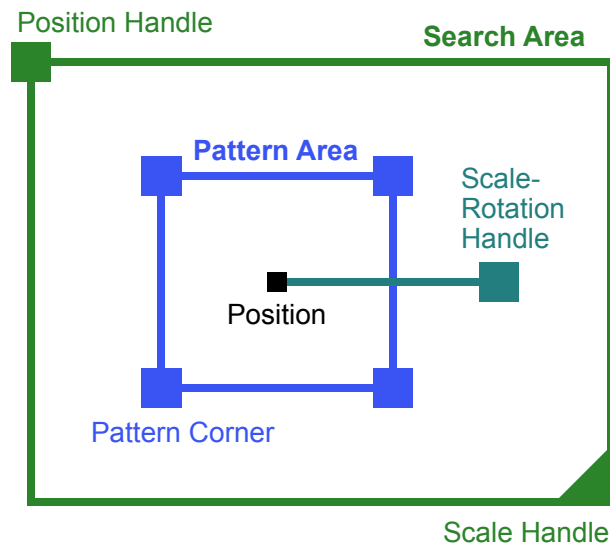
When a clip is loaded a Timeline is shown at bottom of the Preview. It expands over the full area limited by the animation range. You can move the Playhead by dragging with LMB.

The Timeline is composed of the following visual elements:

- Blue line: Playhead
- Yellow: Motion track
- Yellow line: Keyframe
- Orange line: Shape keyframe
- Purple: Prefetched frames
- Light green line: Solve start/end keyframe

Tracking Marker

Point



Marker schematic.

The whole marker can be moved with RMB or by dragging the anchor point (black dot) with LMB. Pressing G also moves the whole marker. When pressing G twice the marker will be moved while keeping the anchor in place. Note that the anchor point outside the pattern area is shown as a cross connected with marker position with a dashed line.

S scales the whole marker. The whole pattern area only will be scaled by pressing S twice; The Pattern can also be rotated using the R key which, depending on the used pivot point, will either rotate patterns around their own centers or rotate the whole markers around e.g. the median point.

To match the perspective transformation of a marker on a plane, the individual corners must be edited manually. Each marker corner can deform individually to define the shapes. Corner positions can be edited by

dragging them with a mouse. Dragging with LMB will change the position of an individual corner.

Note

Note that deforming a pattern is not only useful for planar / affine tracking. Since only pixels within the pattern will be considered this can help to specify a better pattern to track even for simple position tracking.

The Search area can not be rotated; this is intentional. It doesn't make sense to deform the search area.

Plane

The left bottom corner of the plane does have X/Y axis (X is red, Y is green) to help distinguishing orientation of the plane in space.

It is likely that corner of the plane object need to be manually adjusted. To do this sliding individual corners with mouse LMB or general transform tools G, R, S could be used.

Adjusting plane corners will keep it following the plane defined by tracks it was originally created from.

Toolbar

- [Track](#)
 - [Clip](#)
 - [Marker](#)
 - [Tracking Settings](#)
 - [Track](#)
- [Solve](#)
 - [Plane Track](#)
 - [Solve](#)
 - [Cleanup](#)
 - [Geometry](#)
 - [Orientation](#)
 - [Scene Setup](#)

Track

Clip

Set Scene Frames

See [Set Scene Frames](#).

Prefetch

See [Prefetch](#).

Reload

See [Reload Clip](#).

Marker

Add

See [Add Marker](#).

Delete

See [Delete Track](#).

Detect Features

See [Detect Features](#).

Tracking Settings

This panel contains all settings for the 2D tracking algorithms. Preset tracking settings can be configured in the panel header. These presets are based on tracking experience of real footage and provides good start values to begin working with a specific footage.

Pattern Size, Search Size

Defines size of a newly created tracks.

Motion Model

Defines which possible motions tracking feature has. This option should be set depending on which motion a particular feature has and it will make tracking most accurate for such a motion.

Location, Location & Rotation, Location & Scale, Location, Rotation & Scale, Affine

Perspective

Is usually used to track a planar feature, but often *Affine* is a good enough approximation and may have more stable tracks.

Match

Controls which patterns get tracked; to be more precise, the pattern from which frame is getting tracked. Here is an example which should make things clearer.

The tracker algorithm receives two images inside the search area and the position of a point to be tracked in the first image. The tracker tries to find the position of that point from the first image in the second image.

Now, this is how tracking of the sequence happens. The second image is always from a frame at which the position of marker is not known (next tracking frame). But a different first image (instead of the one that immediately precedes the second image in the footage) can be sent to the tracker.

Keyframe

An image created from a frame on which the track was keyframed. This configuration prevents sliding from the original position (because the position which best corresponds to the original pattern is returned by the tracker), but it can lead to small jumps and can lead to failures when the feature point is deformed due to camera motion (perspective transformation, for example).

Previous Frame

Keyframes for tracks are creating every frames, and tracking between keyframed image and next image is used. In this configuration the pattern is tracking between two neighboring frames. It allows dealing with cases of large transformations of the feature point but can lead to sliding from the original position, so it should be controlled.

Prepass

Enables a two pass tracking, where the first pass is a brute force tracking of location only, and the second pass will use tracking of the full motion model refining the first pass.

Normalize

Means patterns will be normalized by their average intensity while tracking, to make them invariant to illumination changes. An example where this is useful is a scene where a marker moves in the shadow of an object.

R, G, B

Defines color channels which will be used by a tracking algorithm. Disabling some colors might increase the contrast to enhance the feature detection.

Copy from Active Track

Copies all settings from active track. Allows to ease creation of new tracks with the same setting.

Tracking Extra Settings

Weight

See Track [Weight](#).

Correlation

This value defines the minimal correlation between a matched pattern and a reference to be considered a successful tracking. If the tracker is stops too early, decrease this value, or if the track is slipping too much when it should stop sooner, increase this value.

Margin

Can be used to disable tracks when they become too close to the image boundary. This slider sets “too close” in pixels.

Use Mask

Allows to use annotation tool to mask part of a pattern, narrowing down what the tracker algorithm is attempting to match across frames.

Frames Limit

Controls how many frames can be tracked when the Track Sequence operator is called. So, each Track Sequence operation would track maximum *Frames Limit* frames. This also helps to notice a slide-off of tracks and correct them.

Speed

Marker settings only – Can be used to control the speed of sequence tracking. This option does not affect the quality of tracking; it just helps to control if tracking happens accurately. In most cases tracking happens much faster than real-time, and it is difficult to notice when a track began to slide out of position. In such cases *Speed* can be set to Double or Half to add some delay between tracking two frames, so a slide-off would be noticed earlier and the tracking process can be canceled to adjust positions of tracks.

Track

Track

See [Track Motion](#).

Clear

See [Before](#).

Refine

See [Refine](#).

Merge

Join Tracks

See [Join Tracks](#).

Solve

Plane Track

See [Create Plane Track](#).

Solve

Tripod

Tripod Motion can be used for footage where the camera does not move and only rotates. Such footage can't be tracked with a generic solver approach, and it is impossible to determine the actual feature points in space due to a lack of information. So this solver will solve only the relative camera rotation and then reproject the feature points into a sphere, with the same distance between feature and camera for all feature points.

Note

This is special type of camera solver and it behaves different from regular solver. It means using more tracks doesn't imply more accurate solution. Having 5-10 tracks on frame is likely what shall be commonly used for this kind of solver.

Keyframe

Automatically select keyframes for initial reconstruction. This option enables complex algorithms which tries to find a keyframe pair with minimal reconstruction error and best scene scale guess.

Keyframe A/B

Start (A) and End (B) frame of the range used for reconstruction.

Refine

Specifies which parameters should be refined during solve. Such refining is useful when you are not sure about some camera intrinsics, and solver should try to find the best parameter for those intrinsics. But you still have to know approximate initial values – it will fail to find correct values if they were set completely incorrectly initially.

Focal Length

Refine the camera's [Focal Length](#).

Optical Center

Refine the camera's [Optical Center](#).

Radial Distortion

Refine the camera's [Radial Distortion Parameters](#).

Tangential Distortion

Refine the camera's [Tangential Distortion Parameters](#).

Solve Camera/Object Motion

See [Solve Solution](#).

Cleanup

This panel contains operators and their settings which are needed to clean up bad tracks: tracks which are not tracked long enough or which failed to reconstruct accurately.

Frames

Tracks or tracked segments shorter than this number of frames will be removed.

Error

Tracks which has reprojection error higher than this value will be removed.

Type

Several actions can be performed for bad tracks:

Select

They can simply be selected.

Delete Track

The whole track can be deleted.

Delete Segments

Bad segments of tracked sequence can be removed.

Clean Tracks

See [Clean Tracks](#).

Filter Tracks

See [Filter Tracks](#).

Geometry

3D Markers to Mesh

See [3D Markers to Mesh](#).

Link Empty to Track

See [Link Empty to Track](#).

Orientation

Scene orientation tools can be used for orienting object to bundles.

Floor

See [Set Origin](#).

Wall

See [Set Floor](#).

Set Origin

See [Set Floor](#).

Set X, Y Axis

See [Set X/Y Axis](#).

Set Scale

See [Set Scale](#).

Apply Scale

See [Apply Solution Scale](#).

Distance

Distance in active scene units which is used by Set/Apply scale.

Scene Setup

Set as Background

See [Set as Background](#).

Setup Tracking Scene

See [Setup Tracking Scene](#).

Selecting

All A

Selects all items.

None Alt-A

Deselects all items.

Inverse Ctrl-I

Selects non-selected items and deselects existing selection.

Box Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Box Select</i>
Shortcut::	B

See [Select Box](#).

Circle Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Circle Select</i>
Shortcut::	C

See [Select Circle](#).

Lasso Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Lasso Select</i>
Shortcut::	Ctrl-Alt-RMB

See [Select Lasso](#).

Select Grouped

Reference

Mode::	All modes
Menu::	<i>Select ▶ Select Grouped</i>
Shortcut::	Shift-G

Select all tracks from specified group.

Action

The group of tracks to select.

Keyframed Tracks::

Select all keyframed tracks.

Estimated Tracks:: Select all estimated tracks.

Tracked Tracks:: Select all tracked tracks.

Locked Tracks:: Select all locked tracks.

Disabled Tracks:: Select all disabled tracks.

Track with Same Color::

Select all tracks with same color as active track.

Failed Tracks:: Select all tracks which failed to be reconstructed.

Select Stabilization Tracks

Reference

Mode:: Tracking mode
Menu:: *Select ▶ Select Stabilization Tracks*

Select tracks which are used for translation stabilization.

Select Stabilization Rotation Tracks

Reference

Mode:: Tracking mode
Menu:: *Select ▶ Select Stabilization Rotation Tracks*

Select tracks which are used for rotation stabilization.

Editing

- [Clip](#)
 - [Open Clip](#)
 - [Set Scene Frames](#)
 - [Set Principal to Center](#)
 - [Prefetch](#)
 - [Reload Clip](#)
 - [Proxy](#)
 - [Set as Background](#)
 - [Setup Tracking Scene](#)
- [Track](#)
 - [Transform](#)
 - [Track Motion](#)
 - [Clear](#)
 - [Refine](#)
 - [Add Marker](#)
 - [Detect Features](#)
 - [Create Plane Track](#)
 - [Solve Solution](#)
 - [Join Tracks](#)
 - [Average Tracks](#)
 - [Copy Tracks](#)
 - [Paste Tracks](#)
 - [Animation](#)
 - [Show/Hide](#)
 - [Clean Up](#)
 - [Delete Track](#)
 - [Delete Marker](#)
- [Reconstruction](#)
 - [Set Origin](#)
 - [Set Floor](#)
 - [Set Wall](#)
 - [Set X/Y Axis](#)
 - [Set Scale](#)

- [Apply Solution Scale](#)
- [Link Empty to Track](#)
- [3D Markers to Mesh](#)

Clip

Open Clip

Reference

Mode:: All modes
Menu:: *Clip ▶ Open Clip*
Shortcut:: Alt-0

Todo.

Set Scene Frames

Reference

Mode:: Tracking
Menu:: *Clip ▶ Set Scene Frames*

Sets end scene frame to match current clip duration.

Set Principal to Center

Reference

Mode:: Tracking
Menu:: *Clip ▶ Set Principal to Center*

Changes the [Optical Center](#) values to the center of image.

Prefetch

Reference

Mode:: All modes
Menu:: *Clip ▶ Prefetch*
Shortcut:: P

Fills cache with frames. As many frames as fits into cache are load form the drive. This allows to fill in the cache as fast as possible when you really need to track something, but this keeps CPU and drive bandwidth idle if you've got a Clip editor opened but not actually interacting with it.

Reload Clip

Reference

Mode:: All modes
Menu:: *Clip ▶ Open Clip*

Force reload the currently loaded movie clip. Is mainly useful when the clip gets edited outside of Blender.

Proxy

Todo.

Set as Background

Reference

Mode:: Tracking
Menu:: *Clip ▶ Set as Background*

Sets the clip currently being edited as the camera background for all visible 3D Viewports. If there is no visible 3D Viewports or the Clip Editor is open

in full screen, nothing will happen.

Setup Tracking Scene

Reference

Mode:: Tracking

Menu:: *Clip ▶ Setup Tracking Scene*

Performs all usual steps to set up a VFX scene:

- Create reference objects for floor and test object.
- Create node set up for combining CG with an actual clip.

Track

Transform

Todo.

Track Motion

The Track Motion menu is used to perform tracking of selected tracks (i.e. following the selected feature from frame to frame).

This operator depends on settings from the Tracking Settings panel. If during sequence tracking the algorithm fails to track some markers, they will be disabled and tracking will continue for the rest of the markers. If the algorithm fails when tracking frame-by-frame, the marker is not disabled, and the most likely position of the feature on the next frame is used.

Backwards

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Track Motion ▶ Backwards</i>
Shortcut::	Shift-Ctrl-T

Tracks the motion backward along the sequence.

Frame Backwards

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Track Motion ▶ Frame Backwards</i>

Shortcut:: Alt-Left

Tracks the motion backward by one frame.

Forwards

Reference

Mode:: Tracking

Menu:: *Track ▶ Track Motion ▶ Forwards*

Shortcut:: Ctrl-T

Tracks the motion forward along the whole sequence.

Frame Forwards

Reference

Mode:: Tracking

Menu:: *Track ▶ Track Motion ▶ Frame Forwards*

Shortcut:: Alt-Right

Tracks the motion forward one frame.

Clear

Before

Reference

Mode:: Tracking

Menu:: *Track ▶ Clear ▶ Before*

Shortcut:: Shift-T

Deletes all tracked and keyframed markers after the current frame for all selected tracks.

Clear Active

Limits clear action to only active track (as opposite to all selected ones).

After

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Clear ▶ After</i>
Shortcut::	Alt-T

Deletes all tracked and keyframed markers before the current frame for all selected tracks.

Clear Active

Limits clear action to only active track (as opposite to all selected ones).

Track Path

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Clear ▶ Track Path</i>
Shortcut::	Shift-Alt-T

Clears all markers except the current one from all selected tracks.

Clear Active

Limits clear action to only active track (as opposite to all selected ones).

Clear Solution

Todo.

Refine

This operator will run a tracker from previous keyframe to current frame for all selected markers. Current markers positions are considering initial position guess which could be updated by a tracker for better match.

Useful in cases when feature disappears from the frame and then appears again. Usage in this case is the following:

- When feature point re-appeared on frame, manually place marker on it.
- Use Refine Markers operation to allow tracker to find a better match.

Depending on direction of tracking use either *Forwards* or *Backwards* refining. Accordingly if tracking happens forwards, use *Refine Forwards*, otherwise use *Refine Backwards*.

Backwards

Reference

Mode:: Tracking
Menu:: *Track ▶ Refine ▶ Backwards*

Refine the track backwards.

Forwards

Reference

Mode:: Tracking
Menu:: *Track ▶ Refine ▶ Forwards*

Refine the track forwards.

Add Marker

Reference

Mode:: Tracking
Menu:: *Track ▶ Add Marker*

Places a new marker at the position of the mouse (which is under the button in this case, not ideal but it is just how things work) and then it can be moved to the needed location. When it is moved to the desired position, LMB can be used to finish placing the new marker. Also, Return and Spacebar can be used to finish placing the marker. But it is faster to use Ctrl-LMB to place markers directly on the footage. This shortcut will place the marker in the place you have clicked.

In addition to this until you have released the mouse button, you can adjust the marker position by moving the mouse and using the track preview widget to control how accurately the marker is placed.

Detect Features

Reference

Mode:: Tracking
Menu:: *Track ▶ Detect Features*

Detects all possible features on the current frame and places markers at these features. This operator does not take other frames into account, so it might place markers on features which belong to moving objects. If the camera is turning away from this shot, no markers could be present within the frames after the camera moved away.

There are several properties for this operator:

Placement

Controls where to place markers.

Whole Frame

Places markers throughout the whole frame.

Inside Annotated Area

Places markers inside the area outlined with the [Annotation Tools](#). This can be used to outline some areas with interesting features and place markers only inside the outlined area.

Outside Annotated Area

Places markers outside the area outlined with the [Annotation Tools](#). This can be used to outline areas of no interest (like trees, humans, etc.) and place markers outside of these areas.

Margin

Controls the distance from the image boundary for created markers. If markers are placed too close to the image boundary, they will fail to track really quickly and they should be deleted manually. To reduce the amount of manual clean-up, this parameter can be used.

Threshold

Limits minimal threshold for placing markers. This value comes from the feature detection algorithm and it means: low values means most probably this feature would fail to track very soon, high value means it is not much such track. Amount of markers to be added can be controlled with this value.

Distance

Defines the minimal distance between placed markers. It is needed to prevent markers from being placed too close to each other (such placement can confuse the camera solver).

Create Plane Track

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Create Plane Track</i>

The *Create Plane Track* operator creates a new plane track. Planar tracking takes advantage of the fact that there are often planar surfaces in footage, by attaching markers to points on these flat planes. It can be used to replace things like billboards and screens on the footage with another image or video. It also might be used for masking.

This button will create a plane object which is deforming in the same way as plane defined by all selected point tracks. At least four feature points tracked across the footage which belongs to the plane you want to replace are needed. More tracks will give better estimation of plane motion.

Feature points used to estimate plane motion could be used from any place on the plane, meaning it's not necessarily need to be corners. Corners are not always easy to be tracked, they might be occluded. In this case you can position tracked features that lay on the same plane far away from the actual plane which should be replaced.

This provides more information about the possible deformation of the marker in following frames, and such markers can be tracked even if partially occluded (appear and disappear during the time). It is only required that two neighbor frames have at least four common tracks.

An image can be projected onto the plane with the [Plane Track Deform Node](#) compositing node.

Solve Solution

Reference

Mode:: Tracking
Menu:: *Track ► Solve Solution*

The *Camera Motion* operator solves the motion of camera using all tracks placed on the footage and two keyframes specified on this panel. There are some requirements:

- There should be at least eight common tracks on the both of the selected keyframes.
- There should be noticeable parallax effects between these two keyframes.

If everything goes smoothly during the solve, the average reprojection error is reported to the information space and to the Clip editor header.

Reprojection error means the average distance between reconstructed 3D position of tracks projected back to footage and original position of tracks. Basically, reprojection error below 0.3 means accurate reprojection, (0.3 - 3.0) means quite nice solving which still can be used. Values above 3 means some tracks should be tracked more accurately, or that values for focal length or distortion coefficients were set incorrectly.

Join Tracks

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Join Tracks</i>
Shortcut::	Ctrl-J

This operator joins all selected tracks into one. Selected tracks should not have common tracked or keyframed markers at the same frame.

Average Tracks

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Average Tracks</i>

The Average Tracks operator creates a new tracking marker by averaging the data from the selected tracks. This can be used to improve stability of tracking on blurry or non-very-sharp feature shapes. The operator takes into

account all [Marker properties](#) however, disabled markers do not affect the averaging.

Gaps in the original tracks will be linearly interpolated, to reduce result track jump. Note that this only applies to gaps “in between”. This means that if a track does not have markers in the beginning or end of it, there is nothing to interpolate with and the resulting track will jump.

Keep Original

When enabled, the selected tracks are *not* deleted.

Copy Tracks

Todo.

Paste Tracks

Todo.

Animation

Todo.

Show/Hide

Todo.

Clean Up

Clean Tracks

Reference

Mode:: Tracking

Menu:: *Track ▶ Clean Up ▶ Clean Tracks*

Identifies all tracks which matches settings from above and performs desired action on them.

Tracked Frames

Tracks or tracked segments shorter than this number of frames will be removed.

Reprojection Error

Tracks which has reprojection error higher than this value will be removed.

Action

Several actions can be performed for bad tracks.

Select

They can simply be selected.

Delete Track

The whole track can be deleted.

Delete Segments

Bad segments of tracked sequence can be removed.

Filter Tracks

Reference

Mode:: Tracking

Menu:: *Track ▶ Clean Up ▶ Filter Tracks*

This operator deletes obviously bad tracks (for example, the ones which are too short). Additionally, it identifies tracks which has suspicious spikes in their motion and selects them.

Delete Track

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Delete Track</i>
Shortcut::	X

Delete all selected tracks.

Delete Marker

Reference

Mode::	Tracking
Menu::	<i>Track ▶ Delete Marker</i>
Shortcut::	Shift-X

Todo.

Reconstruction

Scene orientation tools can be used for orienting object to bundles.

Set Origin

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Set Origin*

Transform camera in a way which makes active track to be moved to a scene origin. Only translation is applied to the camera.

Set Floor

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Set Floor*

Use selected three markers to define a floor. Camera will be transformed in a way which makes the selected markers to be flat (have $Z = 0$).

Set Wall

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Set Wall*

Similar to the floor orientation, but defines a wall (selected tracks are placed onto the XZ plane).

Set X/Y Axis

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Set X/Y Axis*

Transform camera in a way which makes active track to become on X or Y axis. No translation is applied, meaning scene origin which was specified before will be preserved.

Set Scale

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Set Scale*

Scale camera or tracking object in a way which makes distance between two selected tracks match the given value in *Distance*.

Apply Solution Scale

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Apply Solution Scale*

Similar to [Set Scale](#), but actually modifies the tracking data.

Link Empty to Track

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ Link Empty to Track*

Creates new empty in 3D Viewport and appends constraint which parts it to the active track.

3D Markers to Mesh

Reference

Mode:: Tracking

Menu:: *Reconstruction ▶ 3D Markers to Mesh*

Creates a mesh which vertices matches positions of reconstructed tracks. It is required to have motion solved first before using this operator. Only tracks from the current tracking object will be used. The intention of this operator is to give a nice starting point for a manual mesh reconstruction.

Sidebar

- [Track](#)
 - [Track](#)
 - [Objects Panel](#)
 - [Plane Track](#)
 - [Camera](#)
 - [Marker](#)
- [2D Stabilization](#)
 - [Introduction](#)
 - [2D Stabilization Panel](#)
 - [Workflow](#)
- [View](#)
 - [Annotations](#)

Track

- [Track](#)
 - [Track Preview Widget](#)
 - [Further Options](#)
- [Objects Panel](#)
- [Plane Track](#)
- [Camera](#)
 - [Lens](#)
- [Marker](#)

Track

Name

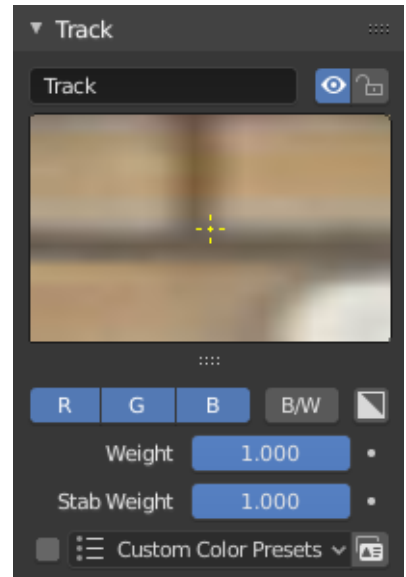
The track name can be changed with this field. Track names are used for linking tracking data to other areas, like a Follow Track constraint.

Enable (eye icon)

This toggle controls if the marker is enabled. If a marker is disabled, its position is not used either by solver nor by constraints.

Lock (padlock icon)

The toggle controls whether the track is locked. Locked tracks cannot be edited at all. This helps to prevent accidental changes to tracks which are “finished” (tracked accurate along the whole footage).



Track panel.

Track Preview Widget

The widget in this panel is called “Track Preview” and it displays the content of the pattern area. This helps to check how accurately the feature is being tracked (controlling that there is no sliding off original position) and also helps to move the track back to the correct position. The track can be moved directly using this widget by mouse dragging.

If an anchor is used (the position in the image which is tracking is different from the position which is used for parenting), a preview widget will display the area around the anchor position. This configuration helps in masking some things when there is no good feature at position where the mask corner should be placed. Details of this technique will be written later.

There is small area below the preview widget which can be used to enlarge the vertical size of preview widget (the area is highlighted with two horizontal lines).

Further Options

R, G, B

Tracking happens in gray-scale space, so a high contrast between the feature and its background yields more accurate tracking. In such cases disabling some color channels can help.

Grayscale Preview (B/W)

Display the preview image as gray-scale even if all channels are enabled.

Mask Preview (black/white icon)

Applies mask defined by an annotation tool in the preview widget.

Weight

When several tracks are used for 3D camera reconstruction, it is possible to assign a reduced weight to some tracks to control their influence on the solution result. This parameter can (and often need to) be animated.

Altering the weights of problem tracking markers can correct or greatly reduce undesirable jumps as feature disappear or become difficult to track.

Another use of Track Weights is when you want to reconstruct a scene from your camera solution. In that case you can first carefully track and solve your scene, and once you are done, lock all your markers with `Ctrl-L`, set the tracker weight in the Extra Settings of the tracker settings to zero and use the feature detection to quickly add lots of markers. Now track them and solve the scene again. Since their weight is zero they will not influence your solution at all, but you will have lots of good reference points in your scene.

Stabilization Weight

While *Weight* parameter is used for 3D reconstruction, the *Stabilization Weight* is used to control 2D stabilization.

Custom Color Presets

The preset for the *Custom Color*.

Custom Color

This setting overrides the default marker color used in the Clip editor and 3D Viewport, and it helps to distinguish different type of features (for example, features in the background vs. foreground and so on).

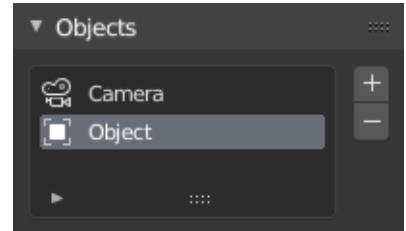
Color also can be used for “grouping” tracks so a whole group of tracks can be selected by color using the Select Grouped operator.

Tip

To select good points for tracking, use points in the middle of the footage timeline and track backwards and forwards from there. This will provide a greater chance of the marker and point staying in the camera shot.

Objects Panel

This panel contains a [list view](#) with all objects which can be used for tracking, camera or object solving. By default there is only one object in this list which is used for camera solving. It cannot be deleted and other objects cannot be used for camera solving; all added objects are used for object tracking and solving only. These objects can be referenced from Follow Track and Object Solver constraints. Follow Track uses the camera object by default.



Objects panel.

If some tracks were added and tracked to the wrong object, they can be copied to another object using *Track ▶ Copy Tracks* and *Track ▶ Paste Tracks*.

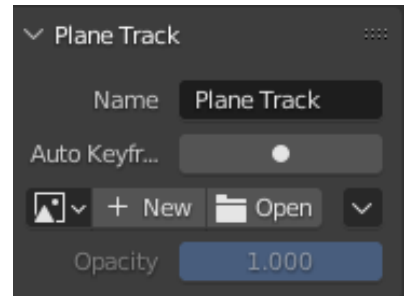
The usage for all kind of objects (used for camera and object tracking) is the same: track features, set camera data, solve motion. Camera data is sharing between all objects and refining of camera intrinsics happens when solving camera motion only.

Plane Track

Its properties are shown only when a plane track is selected.

Name

The name of the selected plane track is shown. It can also be changed from here.



Auto Keyframe

Toggles the auto-keyframing for corners of the plane track. With this enabled, keyframes will automatically get inserted when any corner is moved.

Plane Track panel.

Image

Field to select or create an image which will be displayed inside the plane track. This image is for preview purposes in the Movie Clip editor only. To include it in your final render, see [Plane Track Deform node](#).

New Image from Plane Track

Creates an image from the pixels of the movieclip that the plane marker “sees” at the current frame. This allows to create an unwarped texture of any flat surface in the footage. The resulting image can then be used for editing and retouching, for example to paint out certain parts of the footage.

Update Image from Plane Track

Updates the pixels of the active Plane Track’s image.

Opacity

Used to set the opacity of this image. Again, this is for display purposes only, and will not affect your final render.

Camera

This panel contains all settings of the camera used for filming the movie which is currently being edited in the Clip editor. Different predefined settings can be used here and can be chosen from the panel header. But such settings as distortion coefficients and principal point are not included in the presets and should be filled in even if camera presets are used.

Sensor Width

Is the width of the CCD sensor in the camera. This value can be found in camera specifications.

Pixel Aspect

Is the pixel aspect of the CCD sensor. This value can be found in camera specifications, but can also be guessed. For example, you know that the footage should be 1920×1080, but the images themselves are 1280×1080. In this case, the pixel aspect is: $1920 / 1280 = 1.5$.

Lens

Focal Length

Is self-explanatory; it is the focal length with which the movie was shot. It can be set in millimeters or pixels.

Optical Center

Is the optical center of the lens used in the camera. In most cases it is equal to the image center, but it can be different in some special cases. Check camera/lens specifications in such cases.

Tip

Optical Center also know as the principal point in photogrammetry.

Set Center

See [Set Principal to Center](#).

Lens Distortion

Mathematical function to convert distorted to undistorted coordinates.

Polynomial:: Polynomial radial distortion. Uses three distortion coefficients: K1, K2, and K3.

Division:: It defines high distortions, which makes this model suitable much better for cameras with fisheye lenses. Use two distortion coefficients: K1, K2.

Nuke:: Distortion model used by the Nuke compositor. Use two distortion coefficients K1, K2.

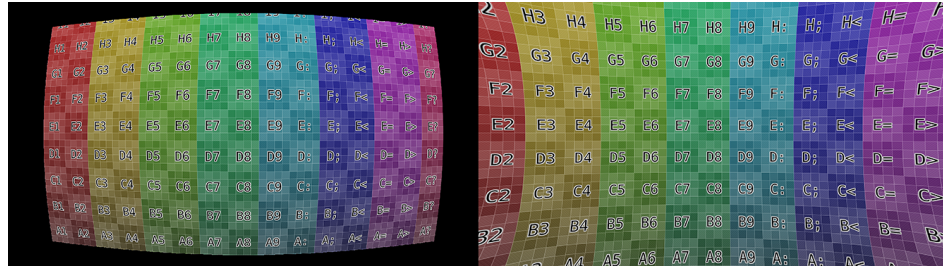
Brown:: Brown-Conrady is one of most advanced mathematical lens distortion models. Used to model both radial and tangential distortion. Can use up to four radial distortion coefficients: K1 - K4 and up to two tangential distortion coefficients: P1 and P2.

Coefficients

Coefficients are used to compensate for lens distortion when the movie was shot. Currently these values can be tweaked by hand only (there are no calibration tools yet) using tools available in Distortion mode. To do this tweak K1 until the solving is the closest to the known focal length (but also take grid and annotations into account to prevent “impossible” distortion).

Radial Distortion Coefficients (K1 - K4)

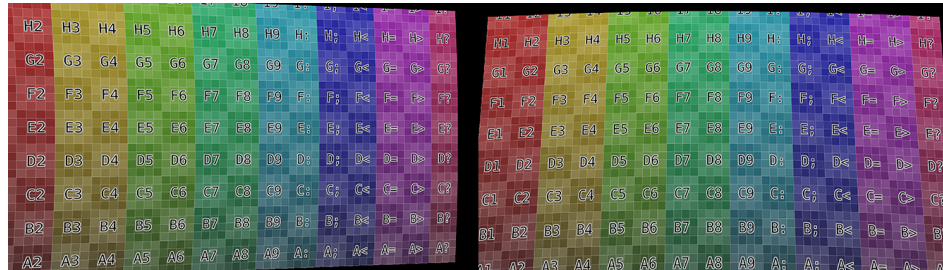
The coefficients in lens distortion models work independent from each other. Positive values will give a barrel distortion while negative values give a pincushion distortion. With a mixture of both negative and positive coefficients you can define more complicated mustache distortions or other complex distortions, that are less common but not rare.



Example of radial distortion for positive and negative K coefficients.

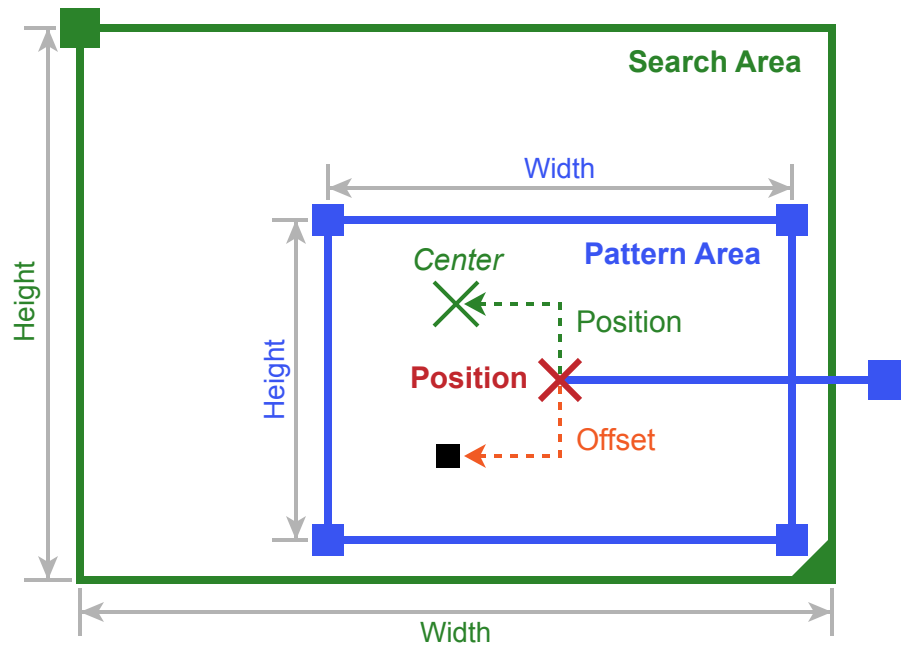
Tangential Distortion Coefficients (P1, P2)

Works independent and allow to compensate for situations when the sensor is not perpendicular to a group of lens. The optical center (also called principal point) will be shifted (distorted) from the center of the sensor. P1 is used to compensate for sensor rotation in Z (vertical) axes, while P2 is for compensating sensor rotation in X (horizontal) axes. Such distortions can be found in sources from cameras with a sensor stabilization system.



Example of tangential distortion for P coefficients.

Marker



Marker schematic.

This panel contains numerical settings for marker position, pattern and search area dimensions, and offset of anchor point from pattern center.

Enabled

Toggles the markers affect on the current frame.

Position X, Y

The X/Y positions of the marker at frame in screen coordinates.

Offset X, Y

The X/Y offset to parenting point.

Pattern Area Width, Height

The width/height of a marker's pattern in screen coordinates.

Search Area X, Y

The X/Y position of search at frame relative to the marker's position.

Search Area Width, Height

The width/height of the markers search in screen coordinates.

2D Stabilization

- [Introduction](#)
 - [How It Works](#)
- [2D Stabilization Panel](#)
 - [Options](#)
- [Workflow](#)
 - [The Simple Case](#)
 - [Avoid Problematic Footage](#)
 - [Elaborate Movements](#)
 - [Animating Stabilization Parameters](#)
 - [Irregular Track Setup](#)

Introduction

The 2D video stabilization is a feature built on top of Blender's image feature tracking abilities: You can use some tracking points to remove shakiness, bumps and jerks from video footage. Typically, image stabilization is part of a 2D workflow to prepare and improve footage prior to further processing or modeling steps. This page helps to understand how it works, introduces related terms and concepts, describes the available interface controls in detail and finally gives some hints about usage in practice.

Typical usage scenarios of the stabilizer:

- Fix minor deficiencies (shaky tripod, jerk in camera movement).
- “Poor man's steadycam” (when a real steadycam was not available, affordable or applicable).
- As a preparation for masking, matching and rotoscoping.

It is not uncommon for 2D stabilization to have to deal with somewhat imperfect and flawed footage.

How It Works

To detect spurious movement in a given shot, we'll assume a simplified model about this movement. We then try to fit the movement of tracked features with this simplified model to derive a compensation. Of course, this works only to the degree our model is adequate – yet in practice, this simplified approach works surprisingly well even with rather complicated shots, where our basic assumption was just an approximation of much more elaborate movements.

This simplified model underlying the 2D stabilization as implemented here assumes movement by an affine-linear transform:

- The camera is pushed up/down/sideways by some translation component.
- The image is then tilted and scaled around a pivot point (rotation center).

To compensate movement according to this simplified model, the 2D stabilizer proceeds in two steps. First we try to detect the translation offset from the weighted average of all *translation* tracking points. After compensating this translation component, we then use additional *rotation/scale* tracking points to detect rotation around a given pivot point. Again, we detect rotation and scale changes through a weighted average of all the rotation/scale tracking points given.

In the current version, the pivot point is anchored to the weight center of the translation tracking points. So effectively the detected translation is already factored out. In some cases this is not optimal, especially when tracks have gaps or do not cover the whole duration of the footage – we plan further options to better control the pivot point in future releases.

Stabilization Tracks

Thus, as foundation for any image stabilization, we need tracked image features to derive the movements. These tracking points or “tracks” can be established with Blender’s [image feature tracking component](#). The right choice of points to track is somewhat tricky, yet crucial for successful image stabilization. Often, we’re here because we’ll have to deal with imperfect footage. In such cases, the averaging of tracks helps to work around image or tracking errors at some point. Moreover, when the footage contains perspective induced movements, symmetrically placed tracking points above and below the horizon can be used to cancel out spurious movement and get stabilization to the focal area in between.



Diverging movements caused by perspective.

Tracks can be added in two groups:

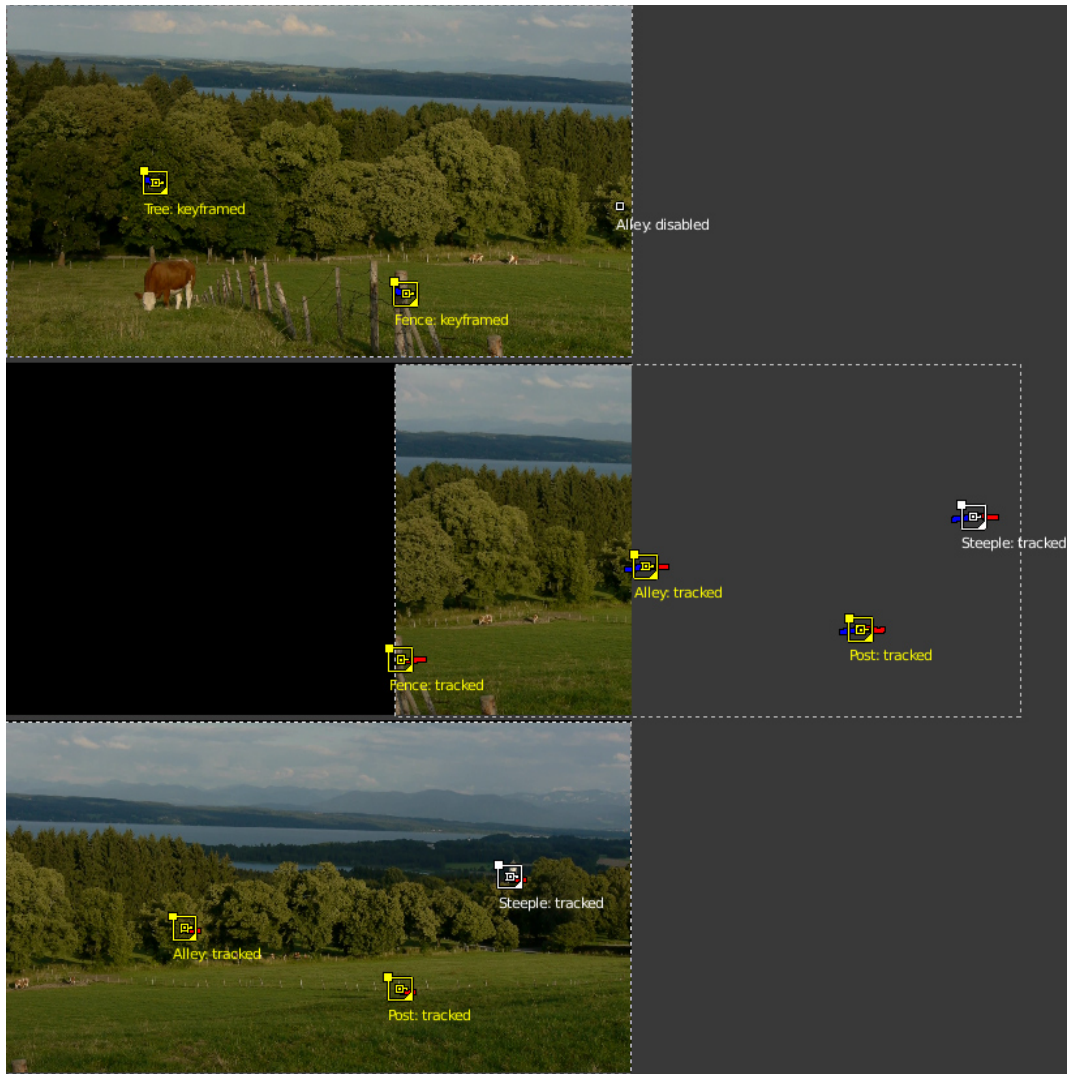
1. First of all is the list of tracks to be used to compensate for jumps in the camera location. From all the tracking points added to this group, we calculate a weighted average. We then try to keep this average location constant during the whole shot. Thus it is a good idea to use tracking markers close to and centered around the most important subject.
2. A second selection of tracks is used to keep the rotation and scale of the image constant. You may use the same tracks for both selections. But usually it is best to use tracking points with large distance from the image center, and symmetrically, on both sides, to capture the angular movements more precisely. Similar to the “location” case, we calculate an average angular contribution and then try to keep this value constant during the whole shot.

Footage, Image & Canvas

When talking about the movement stabilization of video, we have to distinguish several frames of reference. The image elements featured by the footage move around irregularly within the footage’s original image boundaries – this is the very reason why we are using the stabilizer. When our attempt at stabilization was successful, the image elements can be considered stable now, while in exchange the footage’s image boundaries have taken on irregular movement and jump around in the opposite way. This is the immediate consequence of the stabilizer’s activity.

Since the actual image elements, i.e. the subject of our footage can be considered stable now, we may use these as a new frame of reference: we consider them attached to a fixed backdrop, which we call the canvas. Introducing this concept of a “canvas” helps to deal with deliberate movements of the camera. And beyond that, it yields an additional benefit: It is very frequent for the pixels of video footage to be non-square. So we have to stretch and expand those pixels, before we’re able to perform any sensible rotation stabilization. Thus the canvas becomes, by definition, the reference for an undistorted display of the image contents.

But when the camera was moved intentionally, we have to consider yet another frame of reference beyond the canvas: namely the frame (or “cadre”) of the final image we want to create. To understand this distinction, let’s consider a hand-held, panning shot to the right: Since our camera was turned towards the right side, the actual image contents move towards the left side *within* the original image frame. But let’s assume the stabilizer was successful with “fixing” any image contents relative to the canvas – which in turn means, that the original image boundaries start to move irregularly towards the right side, and the contents of the image will begin to disappear gradually behind the left boundary of the original image. After some amount of panning, we’ll have lost all of our original contents and just see an empty black image backdrop. The only solution to deal with that problem is to move the final image frame along to the right, thus following the originally intended panning movement. Of course, this time, we do want to perform this newly added panning movement in a smooth and clean way.



Stabilizing a panning shot.

To allow for such a compensation and to reintroduce deliberate panning, or tilting and zoom of the resulting image, the stabilizer offers a dedicated set of controls: *Expected position*, *Expected rotation* and *Expected scale*. These act like the controls of a virtual camera filming the contents we have fixed onto the canvas. By animating those parameters, we're able to perform all kinds of deliberate camera movements in a smooth fashion.



Restoring the expected camera movement.

The “Dancing” Black Borders

As explained above, when we succeed with stabilizing the image contents, the boundaries of the original footage start to jump around in the opposite direction of the movements compensated. This is inevitable – yet very annoying, since due to the irregular nature of these movements, these “dancing black borders” tend to distract attention from the actual subject and introduce an annoying restlessness. Thus our goal must be to hide those dancing borders as good as possible. A simple solution is to add a small amount of zoom. Sometimes we’ll also need to animate the parameter *Expected* position in order to keep the image centered as good as we can – this helps to reduce the amount of zoom necessary to remove those annoying borders.

The *Autoscale* function can be used to find the minimal amount of zoom just sufficient to remove those black borders completely. However, if the camera jumps a lot, the autoscale function often zooms in too much, especially since this calculation aims at finding a single, static zoom factor

for the whole duration of the footage. When this happens, you'll typically get overall better results with animating both the zoom factor and the expected position manually.

2D Stabilization Panel

The purpose of 2D stabilization is to smooth out jerky camera handling on existing real-world footage. To activate the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Show Stable* in the Clip Display pop-over. Then you'll need to set up some tracking points to detect the image movements.

The 2D Stabilization panel is used to define the data used for 2D stabilization of the shot. Several options are available in this panel: you may add a list of tracks to determine lateral image shifts and another list of tracks to determine tilting and zooming movements. Based on the average contribution of these tracks, a compensating movement is calculated and applied to each frame.

When the footage includes panning and traveling movements, the stabilizer tends to push the image out of the visible area. This can be compensated by animating the parameters for the intentional, “expected” camera movement.

Note

To *activate* the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Show Stable* in the *Clip Display* pop-over.

Options

Anchor Frame

Reference point to anchor stabilization: other frames will be adjusted relative to this frame's position, orientation and scale. You might want to select a frame number where your main subject is featured in an optimal way.

Stabilization Type

Rotation

In addition to location, stabilizes detected rotation around the *rotation pivot point*, which is the weighted average of all location tracking points.

Scale

Compensates any scale changes relative to center of rotation.

Tracks for Stabilization

Location

List of tracks to be used to compensate for camera jumps, or location movement.

Rotation/Scale

List of tracks to be used to compensate for camera tilts and scale changes.

Autoscale

Finds smallest scale factor which, when applied to the footage, would eliminate all empty black borders near the image boundaries.

Max

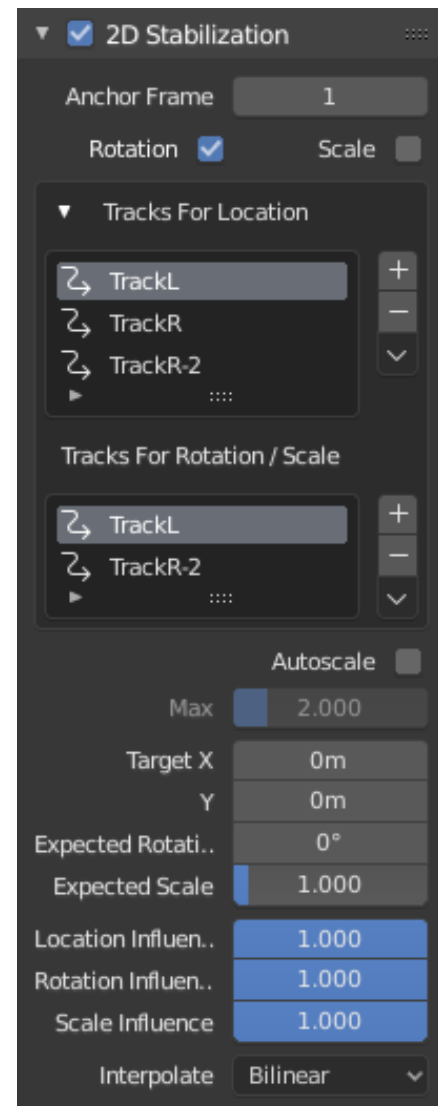
Limits the amount of automatic scaling.

Expected Position X/Y

Known relative offset of original shot, will be subtracted, e.g. for panning shots.

Expected Rotation

Rotation present on original shot, will be compensated, e.g. for deliberate tilting.



2D Stabilization panel.

Expected Zoom

Explicitly scale resulting frame to compensate zoom of original shot.

Influence

The amount of transformation applied to the footage can be controlled. In some cases it is not necessary to fully compensate camera jumps. The amount of stabilization applied to the footage can be controlled. In some cases you may not want to fully compensate some of the camera's jumps. Please note that these “* *Influence*” parameters do control only the *compensation movements* calculated by the stabilizer, not the deliberate movements added through the “*Expected **”-parameters.

Interpolate

The stabilizer calculates compensation movements with sub-pixel accuracy. Consequently, a resulting image pixel needs to be derived from several adjacent source footage pixels. Unfortunately, any interpolation causes some minor degree of softening and loss of image quality.

Nearest

No interpolation, uses nearest neighboring pixel. This setting basically retains the original image's sharpness. The downside is we also retain residual movement below the size of one pixel, and compensation movements are done in 1 pixel steps, which might be noticeable as irregular jumps.

Bilinear

Simple linear interpolation between adjacent pixels.

Bicubic

Highest quality interpolation, most expensive to calculate.

Workflow

Depending on the original footage's properties, achieving good stabilization results might be simple and easy, or it might require some work, dedication and careful planning. This section covers some practical considerations to help improving the results.

The Simple Case

Whenever the camera is basically fixed, or at least “almost” stationary, and the footage is crisp and without motion blur, perfect stabilization is easy to achieve. This might be the case when a tripod was used, but wind or vibrations on the floor (e.g. on a stage) caused some minor shakes. Shoulder camera shots done by an experienced operator also frequently fall into this category.

- Use as few points as possible. Start with a single point right on the main subject.
- Track this single point as accurate as possible. Beware of movements and shape changes of the tracked feature. Proceed in small increments (e.g. 50 frames), zoom in and readjust the target point manually when it drifts away. Another option is to use a larger target area for tracking; since we're tracking only a single point, the slower tracking speed might be acceptable.
- After enabling the basic (location) stabilization, consider if you really need rotation stabilization. Often, some minor, slow swinging movements are not really noticeable and do not warrant the additional working time and quality loss caused by rotation and scale stabilization.
- For rotation, start with one extra point, well spaced but preferably still attached to the main subject.
- Consider to fix some slow residual motion by manually animating the “*Expected **” parameters, before you even think of adding more tracking markers. Because doing so is often not worth the effort.

- If you need to add more points, the most important goal is to achieve *symmetry*. Place location tracking points symmetrically above and below the horizon. Place rotation tracking points into diagonally opposed direction, always centered around the main focal area.

Avoid Problematic Footage

The 2D stabilizer can not work miracles; some flaws simply can not be fixed satisfactory. Notorious issues are motion blur, rolling shutter, pumping autofocus and moving compression artifacts. Especially if you do succeed with basic stabilization, such image flaws become yet the more noticeable and annoying. When on set or on location, it might be tempting to “fix matters in post production”. Resist that deception, it rarely works out well.

- Prefer a short exposure time to avoid motion blur. While motion blur is good to render filmed movements more smooth and natural, it seriously impedes the ability to track features precisely. As a guideline, try to get at least to 1/250 s.
- Prefer higher frame rates. The more *temporal resolution* the stabilizer has to work on, the better the results. If you have the option to choose between progressive and interlaced modes, by all means use interlaced and deinterlace the footage to the *doubled frame rate*. This can be done with the [yadif](#) filter of FFmpeg: use the mode 1 (send_field).
- Beware of the [Rolling Shutter](#) effect. Avoid fast lateral movements. If you can, prefer a camera which produces less rolling shutter. Also, using a higher frame rate reduces the amount of rolling shutter; another reason to prefer interlaced over progressive for the purpose at hand.
- Switch off autofocus. Better plan your movement beforehand, set a fixed focus and rely on depth of field through using a small aperture. Pumping movements might not be so noticeable to the human observer, but the feature tracking tends to slide away on defocused image elements; fixing this manually after the fact can cause a huge waste of time.
- Increase the lighting level, at least use a higher sensitivity. This helps to set a fast shutter speed plus a small aperture. Better lighting and good exposure also help to reduce the impact of compression artifacts. If you can, also select a codec with less data reduction, better color

space, etc. Inevitably, we're losing some quality through the interpolation necessary for stabilization. Plus we're losing some quality due to color space conversion.

Elaborate Movements

When the footage builds on elaborate intended movement of the camera, the process of stabilization becomes more involved – especially when there is a shift in the main area of interest within the shot. When working with many tracks and fine-grained animation, it is easy to get into a situation where additional manipulations actually decrease the quality, while it might be hard to spot and locate the root cause of problems. Recommendation is to proceed systematically, starting from the general outline down to tweaking of specific aspects.

1. Understand the nature of the movements in the shot, both the intended and the accidental.
2. Track some relevant features for location.
3. Establish the basic location stabilization. This includes the decision, which feature to use for what segment of the shot. Work with the track weights to get an overall consistent movement of the weight center, in accordance with the inherent focus of the shot.
4. Define the panning movements of the virtual camera (through animation of the *Expected Position* parameter).
5. Add tracking for rotation and zoom stabilization.
6. Fine-tuning pass:

Break down the whole duration of the shot into logical segments to define the intended camera movement. Then refine those segments incrementally step-by-step, until the overall result looks satisfactory...

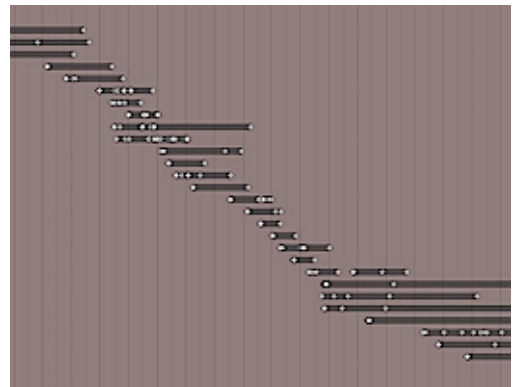
Animating Stabilization Parameters

Animating some parameters over duration of the shot is often necessary, at least to get the final touch, including control of the scale factor to hide the dancing black borders. Unfortunately there is a **known limitation** in the current version: it is not possible to open the generic animation editors (Graph editor and Dope Sheet) for animation data beyond the 3D scene. So, while it *is possible* to set keyframes *right within the UI controls* of the stabilizer (either through pressing the I key or with the help of the context menu), it is not possible to manipulate the resulting curves graphically. The only way to readjust or remove a misguided keyframe is to locate the timeline to the very frame and then use the context menu of the animated UI control. (Hint: the color of the UI control changes when you have located at precisely the frame number of the keyframe.)

Irregular Track Setup

It might not be possible to track a given feature over the whole duration of the shot. The feature might be blurred or obscured; it might even move out of sight entirely, due to deliberate camera movement. In such a situation, we need *another tracked feature* to take on its role, and we need some *overlap time* to get a smooth transition without visible jump.

The stabilizer is able to deal with gaps and partial coverage within the given tracks. However, the basic assumption is that each track covers a single, fixed reference point whenever there is any usable/enabled data. Thus, you must not “reuse” a given track to follow several different points, rather you should disable and thus end one track, when tracking this feature is no longer feasible. You may include “gaps”, when a tracking point is temporarily disabled or unavailable, but you should start a new track for each distinct new feature to be tracked.



Irregular Tracks.

Each track contributes to the overall result by the degree controlled through its *Stab Weight* parameter. It is evaluated on a per-frame basis, which enables us to control the influence of a track by *animating* this *Stab Weight*. You may imagine the overall working of the stabilizer as if each tracking point “drags” the image through a flexible spring: When you turn down the *Stab Weight* of a tracking point, you decrease the amount of “drag” it creates. Sometimes the contribution of different tracks has to work partially counter each other. This effect might be used to cancel out spurious movement, e.g. as caused by perspective. But when, in such a situation, one of the involved tracks suddenly goes away, a jump in image position or rotation might be the result. Thus, whenever we notice a jump at the very frame where some partially covered track starts or ends, we need to soften the transition. We do so by animating the *Stab Weight* gradually down, so that it reaches zero at the boundary point. In a similar vein, when we plan a “handover” between several partially covered tracks, we define a *cross-fade* over the duration where the tracks overlap, again by animating the *Stab Weight* parameters accordingly. But even with such cross-fade smoothing, some residual movement might remain, which then needs to be corrected with the *Expected Position* or *Expected rotation* parameters. It is crucial to avoid “overshooting” movements in such a situation – always strive at setting the animation keyframes onto precisely the same frame number for all the tracks and parameters involved.

View

Annotations

Annotation tool strokes can be enabled/disabled with the checkbox in the panel header. It is a standard annotations panel where annotation layers and frames can be controlled. There is one difference in the behavior of the annotation tools from other areas – when a new layer is created “on-demand” (when making a stroke without adding a layer before this) the default color for the layer is set to pink. This heightens the color contrast to make the stroke more noticeable on all kinds of movies.

Data Source

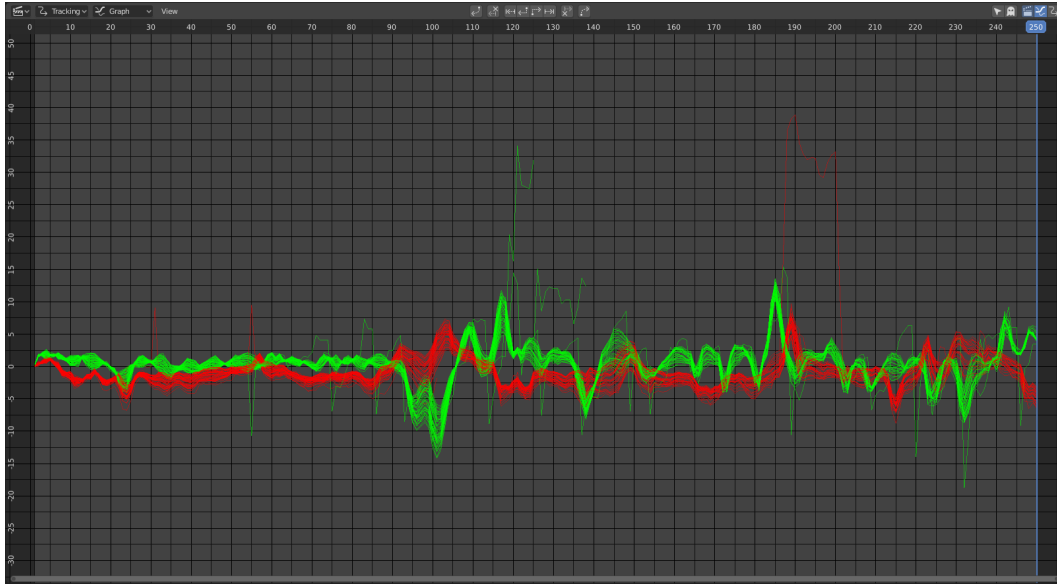
Determines the data-block type the current annotation layer is stored.

Clip::	Store the current annotation layer with the active <i>Movie Clip</i> data-block.
Track::	Store the current annotation layer with the active <i>Track</i> data-block.

See also

[Annotation Tools](#) for more information on general annotation layers.

Graph View



Graph View.

Introduction

The graph or curves view has numerous purposes based on the color of the lines. The red and green lines on the graph show you the speed of the trackers at a given frame. Green is vertical movement, Red is horizontal. Therefore the first frames will always be at zero.

The blue line is the line that comes out when you click on the film strip is the average per-frame error. This curve is available only after pressing camera solve and is not editable. This is the one line that you want to be as flat as possible and as closer to zero as you can. The high points will show you where in your shot you are having inaccurate tracking.

Frames outside of scene frame range are darkened.

Header

Show Selected (mouse cursor icon)

Displays the graph for only selected trackers.

Display Hidden (ghost icon)

Displays channels from objects that are hidden.

Filter

Display options, defines what curves are visible.

Frames

Visualizes per-frame average reprojection error of all tracks in the active tracking object.

Motion

Shows curves for X and Y speed of tracks.

Error

Per-frame reprojection error of tracks.

Usage

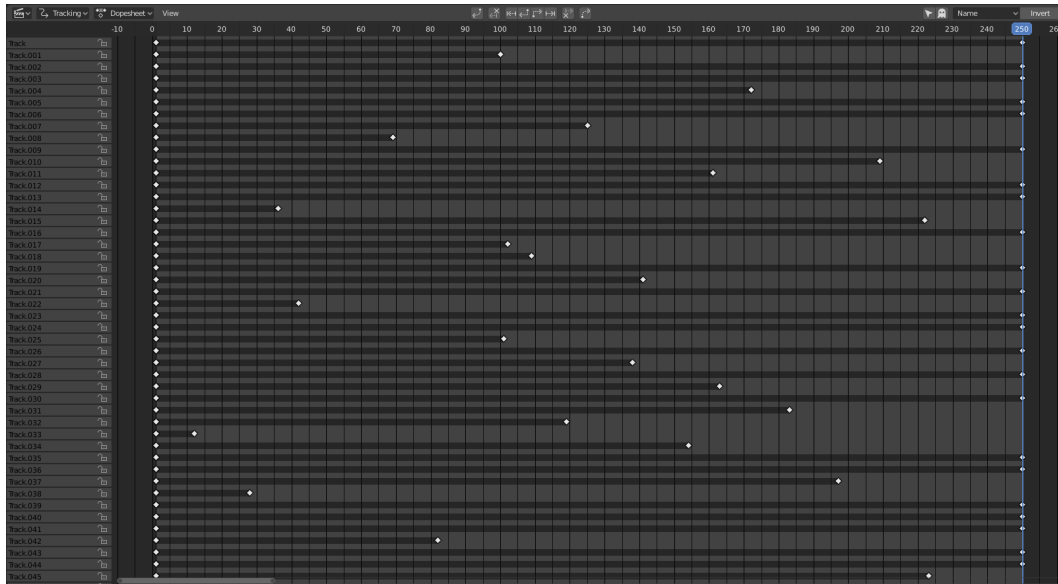
The curves are useful to see if particular trackers are moving differently than the average. A line that spikes from the rest of the curve usually means a tracking error.

You can manually edit the curve by selecting a point in the curve and dragging it or deleting, that will affect the corresponding tracker on that particular frame.

Lock to Selection L

Locks the view to selected markers during playback.

Dope Sheet View



Dope Sheet View.

The Dope Sheet View is used to visualize motion tracking data, it is implemented as separate view of the Movie Clip editor just like the Graph View.

It displays channels for selected tracks and each channel visualizes tracked segments of tracks as dark bars and keyframed positions of tracks as small diamonds.

The background is highlighted depending on the number of tracks in a frame. This means that if for a frame (or sequence of frames) there are less than eight tracks, the background will turn red; if there are from eight to sixteen tracks, the background will be yellow.

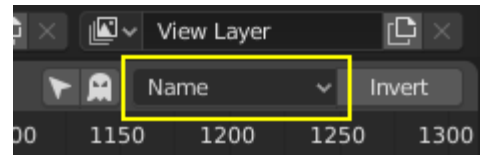
This is only a visual feedback, which doesn't mean that the camera motion will not reconstruct with less than eight tracks. It only means that you should pay attention to those frames and check if all possible good feature points are tracked there. Remember, if there are no good feature points in

the frame and there are less than 16 tracks in the frame, it doesn't mean the solution won't be accurate. Rather, adding more tracks on bad feature points will reduce the accuracy of solution.

Header

Show Only Selected (mouse cursor icon)

Limits Dope Sheet channels to only information about selected tracks.



Hidden (ghost icon)

Includes information from hidden tracks.

Sort order of the channels.

Sort Method

Sort order of the tracks.

Name::	Sort selected tracks in alphabetical order based on their names.
Longest::	Sort tracks by longest tracked segment length.
Total::	Sort tracks by overall amount of frames.
Average Error::	Sort tracks by their average reprojection error after solving camera or object motion.
Start Frame::	Sort channels by first frame number.
End Frame::	Sort channels by last frame number.

Invert

To change the sort order from ascending to descending.

Usage

The Dope Sheet View is for visualization and does not have any tools to actually edit data.

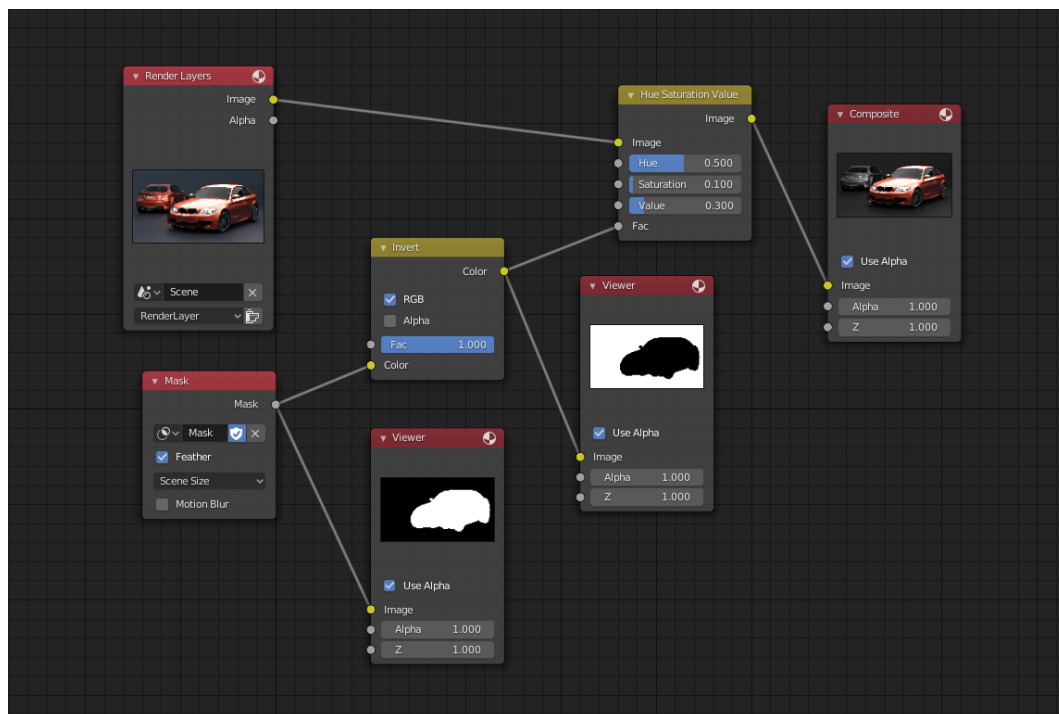
Masking

- [Introduction](#)
 - [Mask Data-block](#)
 - [Header](#)
- [S-Curves](#)
 - [Primitives](#)
- [Selecting](#)
 - [All](#)
 - [None](#)
 - [Invert](#)
 - [Box Select](#)
 - [Circle Select](#)
 - [Lasso Select](#)
 - [Select Linked](#)
- [Editing](#)
 - [Transform](#)
 - [Clear Feather Weight](#)
 - [Toggle Cyclic](#)
 - [Set Handle Type](#)
 - [Recalculate Handles](#)
 - [Switch Direction](#)
 - [Copy Paste](#)
 - [Clear Parent](#)
 - [Make Parent](#)
 - [Animation](#)
 - [Show/Hide](#)
 - [Delete](#)
 - [Miscellaneous](#)
- [Sidebar](#)
 - [Mask Settings](#)
 - [Mask Layers](#)
 - [Active Spline](#)
 - [Active Point](#)

Introduction

Masks can be created in the *Image* and *Movie Clip* editors, by changing the mode to *Mask* in the header. This will add various tools and properties to the editor panels, while hiding others that are not needed for interacting with masks.

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green-screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the Compositor.



Using the Mask node to isolate an object in compositing.

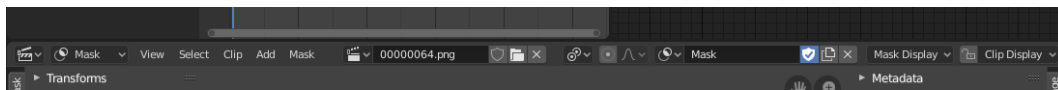
While the *Movie Clip Editor* and *Image Editor* are used to edit masks, the *Compositor* and *Sequencer* are just using already created mask.

Masks can be animated over the time so that they follow some object from the footage, e.g. a running actor. This can be achieved with shape keys or parenting the mask to tracking markers.

Mask Data-block

Mask data-block containing multiple mask layers and splines. They are the most high-level entities used for masking purposes. Masks can be reused in different places, and hold global parameters for all the entities they consist of.

Header



The Movie Clip Editor header in Mask mode.

Menus

View

Center View to Cursor

Move the view so that the 2D cursor is at the center of the editor.

Add

Use to add primitive shapes.

Mask

Operators used to [Edit](#) masks.

Controls

Mask

Once set to Mask mode, a mask data-block can be added with a [data-block menu](#). Any image, movie clip, render or compositing result can be used as a backdrop to display masks over.

New + Alt-N

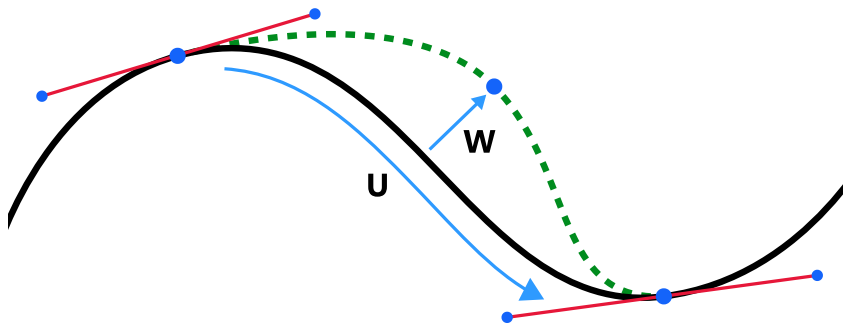
Mask Display

See [Mask Display](#).

S-Curves

The curve type used for creating mask splines is almost a Bézier curve, but with some differences. Smooth edges of the mask are defined by feathering. The curve needed to support feathering in a way that stuck to the curve as you edited it, for ease of editing an animation. These are called S-curves.

Besides the handles, every control point also has points that define the feather between the current point and the next point on the spline. Each feather point is stored in UW space, where U means position across spline segment, and W (weight) means distance between main spline and feather points.



S-Curve explained.

This allows for deforming the main spline in almost any way, and the feather will be updated automatically to reflect that change.

For example if there is just rotation of the spline, feather would stay completely unchanged. If one point's feather is moved, the other feathers will be automatically stretched uniformly along that segment and the overall shape will be almost the same as artists would want it to be.

Primitives

Reference

Mode:: Mask Mode

Tool:: *Add*

Shortcut:: Shift-A

There are two primitives available: a Bézier Circle and a Square with vector handles.

Selecting

All

Reference

Mode::	All modes
Menu::	<i>Select ▶ All</i>
Shortcut::	A

Selects all items.

None

Reference

Mode::	All modes
Menu::	<i>Select ▶ None</i>
Shortcut::	Alt-A

Resets the selection to nothing.

Invert

Reference

Mode::	All modes
Menu::	<i>Select ▶ Inverse</i>
Shortcut::	Ctrl-I

Selects non-selected items and deselects existing selection.

Box Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Box Select</i>
Shortcut::	B

See [Select Box](#).

Circle Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Circle Select</i>
Shortcut::	C

See [Select Circle](#).

Lasso Select

Reference

Mode::	All modes
Menu::	<i>Select ▶ Lasso Select</i>
Shortcut::	Ctrl-Alt-LMB

See [Select Lasso](#).

Select Linked

Reference

Mode::	All modes
Menu::	<i>Select ▶ Select Linked</i>
Shortcut::	Ctrl-L

Select all curve points linked to already selected ones.

Editing

The tools and panels available to edit masks are the same in both editors. Editing of mask splines happens in a way similar to editing Bézier curves or paths in GIMP or other curve editors.

Tip

To get interactive feedback on the resulting mask, a Mask node can be connected directly to a Viewer node in the Compositor, which will then keep updating the compositing result while editing.

Transform

Reference

Mode::	Mask Mode
Menu::	<i>Mask ▶ Transform</i>

[Move](#) G

Change the location of control points. Control points can also be moved with LMB. The whole spline can be moved by dragging the center dot with LMB.

[Rotate](#) R

Change the location of control points by rotating about a pivot point.

[Scale](#) S

Change the location of control points by expanding the distance between points.

[To Sphere](#) Shift-Alt-S

Morphs the control points to the shape of a circle.

[Shear](#) Shift-Ctrl-Alt-S

Shifts control points along a defined axis so parallel control points move past one another.

[Push/Pull](#)

Moves the control points closer together (Push) or further apart (Pull).

Scale Feather Alt-S

Will scale the feather size.

Clear Feather Weight

Reference

Mode:: Mask Mode

Menu:: *Mask ▶ Clear Feather Weight*

Resets the feather weight to zero.

Toggle Cyclic

Reference

Mode:: Mask Mode

Menu:: *Mask ▶ Toggle Cyclic*

Shortcut:: Alt-C

Toggle to create a closed curve or open it again. Close the mask by joining the last control point to the first.

Set Handle Type

Reference

Mode:: Mask Mode

Menu:: *Mask ▶ Set Handle Type*
Shortcut:: *V*

Set handle type for selected spline points.

Recalculate Handles

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Recalculate Handles*
Shortcut:: *Shift-N*

Make normals (handle directions) consistent.

Switch Direction

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Switch Direction*

Switch Direction handle directions in/out.

Copy Paste

Todo.

Clear Parent

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Clear Parent*

Shortcut:: Alt-P

Clears any parenting relationship for the selected spline points.

Make Parent

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Make Parent*
Shortcut:: Ctrl-P

Parents one or more selected spline points to the active motion tracker.

Animation

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Animation*

Masks can be animated with the shape keying system. This can be useful when there are not enough good feature points to track in the footage, or the mask is not based on footage. Mask animation timing can be edited from the *Dope Sheet's* [Mask Mode](#).

Insert Shape Key I

Will insert a shape key for the active mask layer at the current frame. This works on the level of mask layers, so inserting a shape key will keyframe all the splines and points contained in it.

Clear Shape Key Alt-I

Will clear the shape key for the active mask layer at the current frame.

Feather Reset Animation

Resets the feather offset across all animated frames.

Re-Key Points of Selected Shapes

Re-interpolate selected points on across the range of keys selected in the *Dope Sheet*.

Show/Hide

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Show/Hide*

- Hide Selected H
- Hide Unselected Shift-H
- Clear Restricted View Alt-H

Delete

Reference

Mode:: Mask Mode
Menu:: *Mask ▶ Delete*
Shortcut:: X

Removes control points.

Miscellaneous

Slide Spline Curvature LMB

Moves the curve and/or control points by clicking on them and dragging.

Add Vertex and Slide Ctrl-LMB

Inserts new control points and defines handle orientations by a continued mouse drag. If the last point was selected, double-click will also close the curve.

Add Feather Vertex and Slide `Shift-Ctrl-LMB`

Inserts new feather control points that can be transformed independently of the main spline curve. If no feather mask is in use this will create a basic feather mask to the curve.

Sidebar

Mask Settings

Start Frame, End Frame

Set the frame range of the mask for *Sequencer*.

Mask Layers

Mask layers consists of one or several splines and used to “grouped” operation on splines. Layers can be used to create complex shapes and to define how the splines interact with each other. Splines belonging to the same layer can be animated together, for example by an item from motion tracker footage. Example of such tools might be parenting the whole set of splines to single motion tracking data or simple to transform all of them together.

Opacity

Used to set the opacity of the mask layer.

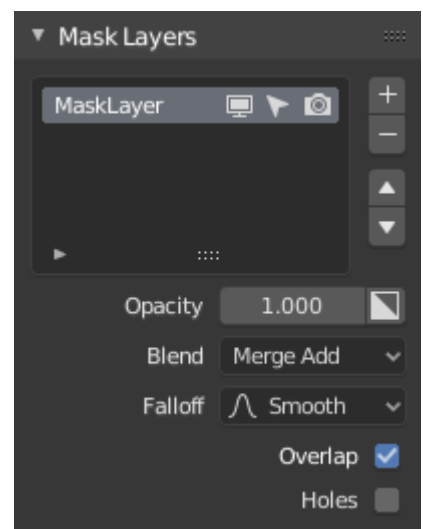
Invert (black/white icon)

Inverts the values (colors) in the mask layer.

Blend

The layer blending operation to perform. See [Color Blend Modes](#).

Modes *Merge Add* and *Merge Subtract* give better results when using a *Feather* on overlapping masks than straightforward mathematical addition and subtraction.



Mask Layer panel.

Falloff

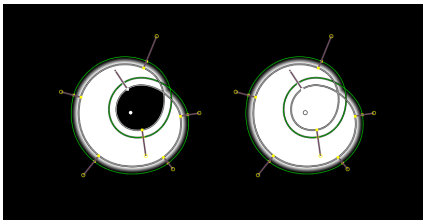
Type of the *Feather* falloff, controls the shape of the transition between black and white.

Overlap

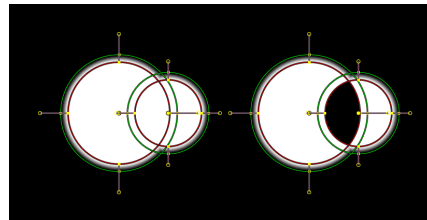
Fills the self-intersecting areas.

Holes

Overlapping splines from the same layer will generate holes in the mask.



The Overlap option example.



The Holes option example.

Example

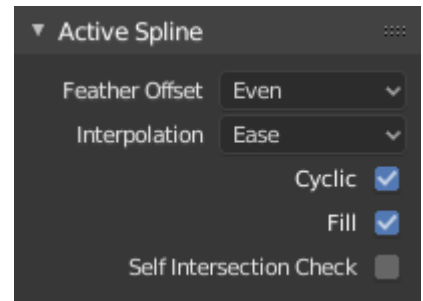
The purpose of mask layers can be explained with an example. Suppose there are two unwanted people in the footage, and one of them goes from left to right, and the other in the opposite direction. Two mask layers can then be used to mask them separately by using a single mask data-block. At the point of intersection of these shapes they will be added together rather than creating a hole, as would happen if they were on the same layer. If the motion is simple enough, a single motion tracked point can be used to drive the location of the entire mask layer.

Active Spline

Feather Offset

The method used for calculating the offset of the mask spline feather.

Even::



Active Spline panel.

Smooth::

Preserves the thickness of the feather, but can give undesirable loops of the feather curve.
Gives a nicer and smoother shape, but can also give an undesirable sharp feather when a curve segment forms an S-shape.

Weight Interpolation

The type of weight (thickness of feather) interpolation between points.
Linear or *Ease* (i.e. changes occur slowly at the beginning and at the end).

Cyclic

If the spline is closed or not.

Fill

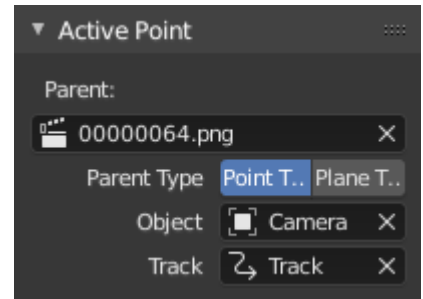
Creates splines with filled areas. If disabled, Blender will create curves with a thickness to mask out thin objects such as wires or hair.

Self Intersection Check

Prevent the feather (not the curve itself) from intersecting with itself.

Active Point

This panel is shown when both a tracking marker and mask is selected.



Active Point panel.

Parent

In the *Movie Clip Editor* it is possible to link the whole mask or its points to motion tracks. This way the mask or points will follow the tracks.

Parent

[Data ID](#) to which the mask or spline is parented to in case of parenting to movie tracking data set to Movie Clip data-block.

Parent Type

Point Track, Plane Track

Object

[Object](#) to parent to.

Track

Name of individual tracks.

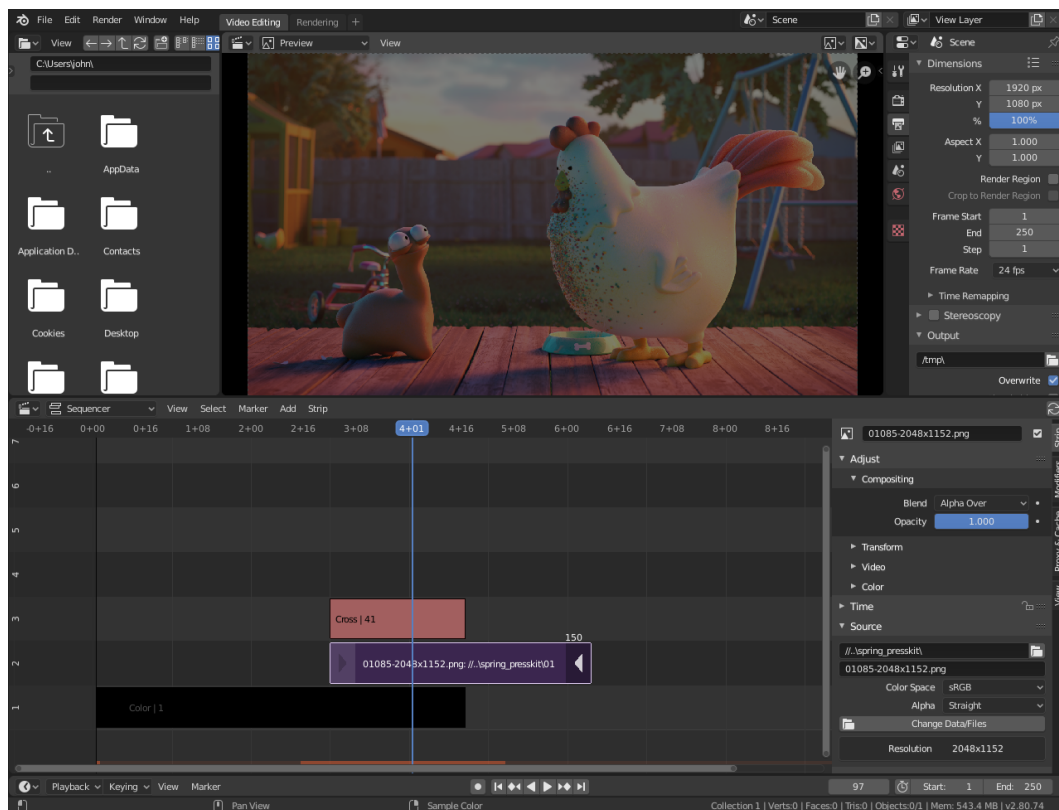
Video Editing

- [Introduction](#)
- [Setup Your Project](#)
 - [Introduction](#)
 - [Directory Structure](#)
- [Edit Your Project](#)
 - [Introduction](#)
 - [Montage](#)

Introduction

In addition to modeling and animation, Blender can be used to edit videos. There are two possible methods for this, one being the [Compositor](#) and the other, described in this chapter, being the Video Sequencer. The Video Sequencer within Blender is a complete video editing system that allows you to combine multiple video channels and add effects to them. You can use these effects to create powerful video edits, especially when you combine it with the animation power of Blender!

To use the Video Sequencer, you load multiple video clips and lay them end-to-end (or in some cases, overlay them), inserting fades and transitions to link the end of one clip to the beginning of another. Finally, you can add audio and synchronize the timing of the video sequence to match it.



Default Video Editing screen layout.

Setup Your Project

- [Introduction](#)
- [Directory Structure](#)

Introduction

The proverb “A good start is half the battle”, certainly applies to video editing. Setting up your *work environment* and *project* to your needs is key to success. In setting up your video project, you have to distinguish between:

- Work Environment related settings and activities:

These settings apply to all of your projects and are set at “Blender level”; for example the installation of add-ons. In fact, they can also influence your non-video editing projects. Most of these settings remain more or less stable throughout your projects and should probably only set once.

- Project related settings and activities:

These settings vary from project to project and are very specific for your project; for example the output media format. For each new project, you have to evaluate these settings and activities.

Of course, many settings and activities occur at both levels. For example, automatic proxies can be enabled globally but changed on a per project or even strip basis. The layout of the Video Editing Workspace is defined at Blender level but can be tweaked per project.

Directory Structure

A video project is most likely a combination of several different assets. You can separate them into three broad categories.

- Video files: video clips (or movies in Blender-talk), photos, graphic files (charts, logos, ...), and Visual Effects (VFX) such as masks, lens flares, animation.
- Audio files: recorded dialog, voice-over, music, and Sound Effects (SFX) such as environmental sounds, swooshes, ...
- Project files: the blend files and backups, (partial) render results, documentation such as scripts and storyboards.

Together they can form rapidly an intangible heap of files. It's good practice to collect all those assets in *one* project directory with appropriate subdirectories. Why? It will lower the probability that you accidentally delete a file (which will result in a 'file not found' error) or that you forget to include a necessary file when transferring the project ('file is missing' error). And most of all, an appropriate directory structure, will help you to keep a clear overview of your assets.

See also

Blender can incorporate some files within the Blend file (see [Packed Data](#)). However, this doesn't work with video files, which can have very huge file sizes. So, it's better to assure that your project directory contains all necessary files.

It's also good practice to use some kind of naming convention and add metadata. Figure 1 shows a possible example, based on our categorization of the assets above. The directories are numbered so that they mimic a normal workflow.

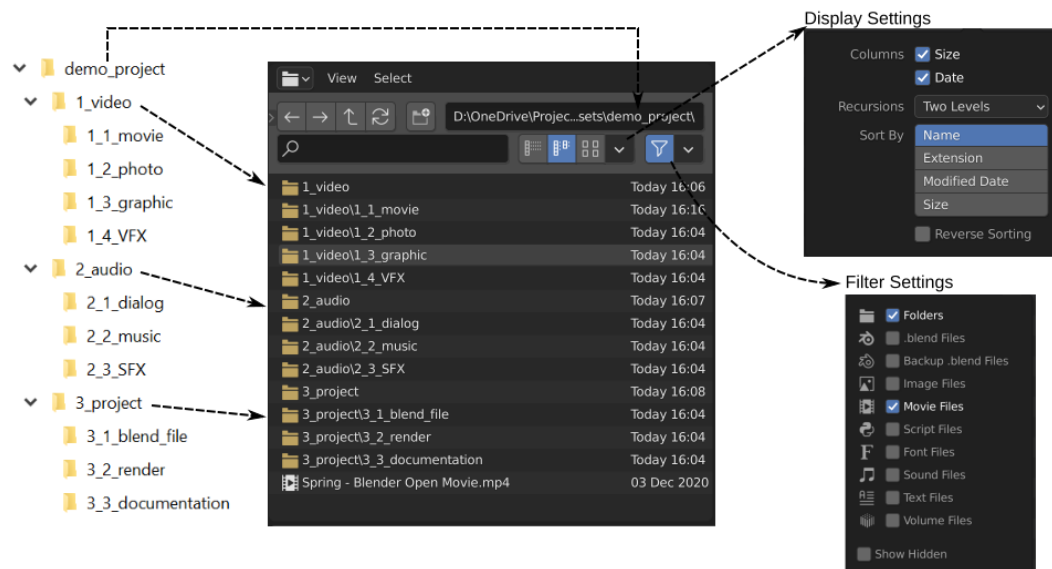


Figure 1: Organizing your project.

Edit Your Project

- [Introduction](#)
- [Montage](#)
 - [Introduction](#)
 - [Strips](#)
 - [Selecting](#)
 - [Editing](#)
 - [Meta Strips](#)

Introduction

There is no single optimal workflow for editing your video projects that fits every case. Although there certainly are some specific use cases; e.g. tutorial editing, wedding videos, ..., some agreement exists about distinguishing four basic activities.

1. Montage: starting with your raw footage, you'll have to deliver a coherent and fluent succession of strips that tell your story. Clips have to be rearranged, combined, split (cut) and trimmed.
2. Effects: these can be simple transition effects between strips, e.g. fade or full-fledged animations, e.g. rolling end credits.
3. Color grading: because your assembled timeline consists of different shots taken with different cameras under different lighting conditions, colors can vary widely between them. With color grading and color correcting you can harmonize the color perception.
4. Sound: this ranges from adding background music and voice-over to creating special sound effects and using third-party software such as Audacity.

Montage

- [Introduction](#)
- [Strips](#)
 - [Introduction](#)
 - [Types](#)
- [Selecting](#)
 - [Select Menu](#)
- [Editing](#)
 - [Transform](#)
 - [Image Transform](#)
 - [Split](#)
 - [Hold Split](#)
 - [Duplicate Strips](#)
 - [Delete](#)
 - [Update Scene Frame Range](#)
 - [Separate Images](#)
 - [Movie Strip](#)
 - [Effect Strip](#)
 - [Lock/Unlock](#)
 - [Mute/Unmute](#)
 - [Inputs](#)
 - [Context Menu](#)
 - [Fades](#)
 - [Retiming](#)
- [Meta Strips](#)

Introduction

Montage is the technique of assembling separate clips (video, audio, text, effects) into a coherent sequence. The importance of montage was first demonstrated by the Russian filmmaker Lev Kuleshov in the 1910s and 1920s. Famous is the Kuleshov effect: viewers derive more meaning from the interaction of two sequential shots than from a single shot in isolation (see the [Wikipedia article](#) for a nice illustration).

Obviously, the first thing in assembling your timeline is importing or adding strips. There are several ways to do this and each has its own advantages and disadvantages.

Before you can do anything with a strip, you have to select them. There are multiple ways of selecting strips.

Strips can be moved in time (left to right on the X axis) or in the display stack (bottom to top on the Y axis).

Splitting or cutting a strip will create two parts of the strip (before and after the split). There are two variants: Split and Hold Split.

Trimming is the process of removing or adding a portion of the video at its head or tail. This will result in a decrease or increase of the duration of the video.

Grouping is the creating of a meta strip whereby several strips are grouped together.

Hint

Creating a good montage is a time consuming process. Therefore, all basic operations (add, split, trim, ...) have an associated shortcut. It pays off to

learn these shortcuts, even though the same result could be obtained with the menu and/or mouse.

Strips

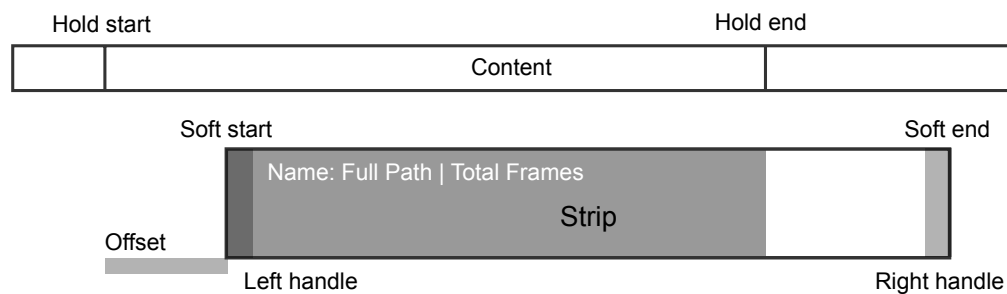
- [Introduction](#)
 - [Adding Strips](#)
 - [Visualization](#)

Types

- [Scene Strip](#)
- [Clip Strip](#)
- [Mask Strip](#)
- [Movie Strip](#)
- [Sound Strip](#)
- [Image/Sequence Strip](#)
- [Color Strip](#)
- [Text Strip](#)
- [Adjustment Layer Strip](#)
- [Effect Strips](#)
 - [Add Strip](#)
 - [Subtract Strip](#)
 - [Multiply Strip](#)
 - [Alpha Over, Under & Over Drop Strips](#)
 - [Color Mix Strip](#)
 - [Multicam Selector Strip](#)
 - [Transform Strip](#)
 - [Speed Control Strip](#)
 - [Glow Strip](#)
 - [Gaussian Blur Strip](#)
- [Transitions](#)
 - [Sound Crossfade](#)
 - [Cross Strip](#)
 - [Gamma Cross Strip](#)
 - [Wipe Strip](#)

Introduction

A strip is a container which carries frames provided by one or more sources (input). It is defined by a *Start Frame* and a *Length*, and is displayed as a colored horizontal rectangle.



Strip schematic.

Adding Strips

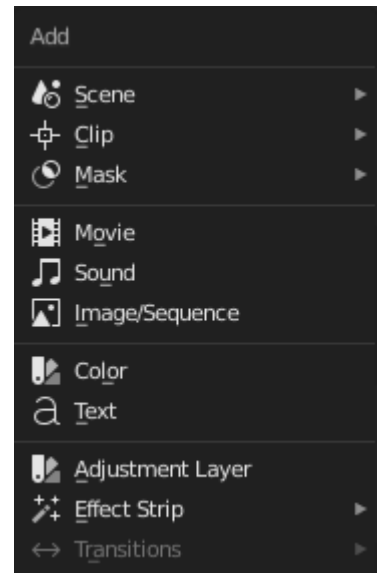
Reference

Menu:: *Add*
Shortcut:: Shift-A

The Add menu is the main menu you will be using to add content to the Video Sequencer. In general, you load up your strips, create strips of special transition effects, and then animate out your sequence by selecting “Do Sequence” and clicking the *Animation* button. You can use the Add menu in the header, or hover your mouse cursor over the Sequence workspace and press Shift-A.

Blender does not care which of these you use; you can freely mix and match any of them. When you choose to add one of these, it lets you either choose a data-block or the editor area will switch to a File Browser for you to select what you want to add. Supported files are filtered by default.

The start frame of the newly created strips will be placed at the position of the frame indicator. When loading multiple files (movie and sound) at the same time each will be added one after the other.



The Add Menu.

Adding Effects & Transitions

Blender offers a set of effects that can be added to your sequence.

To add an effect strip, select one base strip (image, movie, or scene) by LMB clicking on it. For some effects, like the Cross transition effect, you will need to Shift-LMB a second overlapping strip (it depends on the effect you want). From Add menu pick the effect you want. When you do, the Effect strip will be shown above the source strips. If it is an independent effect, like the [Color Generator](#), it will be placed at the position of the frame indicator.

Note

Since most Effects strips depend on one or two source strips, their frame location and duration depends on their source strips. Thus, you may not be able to move it; you have to move the source strips in order to affect the effect strip.

With some effects, like the [Alpha Over](#), the order in which you select the strips is important. You can also use one effect strip as the input or source strip with another strip, thus layering effects on top of one another.

If you picked the wrong effect from the menu, you can always exchange it using [Effect Strip](#).

Visualization

They all become a color-coded strip in the Video Sequencer:

- Scene strip: Light green.
- Clip strip: Dark blue.
- Mask strip: Red.
- Movie strip: Aquamarine.
- Image strip: Purple.
- Sound strip: Turquoise.

Each of the effect strips has its own color.

Besides each of these default colors you can also assign individual strips an alternative color in the [Strip Properties](#).

Note

These colors are dependent on the user interface [Theme](#). The colors described above are in reference to Blender's default theme.

Scene Strip

Scene strips are a way to insert the render output of another scene into your sequence. Instead of rendering out a video, then inserting the video file, you can insert the scene directly.

The strip length will be determined based on the animation settings in that scene.

Note

Scene strips cannot be used to reference the sequence's own scene; a secondary scene must be used instead.

Adding Scene Strips

Existing scenes strips can be added from the *Add ▶ Scene ▶ “Scene Name”*. New scenes can also be created directly from the add menu with *Add ▶ Scene ▶ New Scene*.

Options

Start Frame

The first frame to start the scene strip.

Channel

The channel to place the strip in.

Replace Selection

Replace the active strip with the new scene strip.

When creating a new scene you have the following options:

Type

How the new scene is created.

New::	Add new Strip with a new empty Scene with default settings.
Copy Settings::	Add a new Strip, with an empty scene, and copy settings from the current scene.
Linked Copy::	Add a Strip and link in the collections from the current scene (shallow copy).
Full Copy::	Add a Strip and make a full copy of the current scene.

Options

Scene

A [Data-Block Menu](#) to select or create the scene to render from.

Input

Input type to use for the Scene strip.

Camera::	Use the Scene's 3D camera as input.
Sequencer::	Use the scene's Sequencer timeline as input, allowing one scene to reuse another scene's edit (instead of taking the render output from the scene). This is similar to how Meta Strips work, with the added advantage of supporting multiple instances of the same data.

Camera

This can be used to override the scene's camera with any other object.

It is useful to support switching views within a single scene.

Show

Annotations

Shows [Annotations](#) while in non-render [Preview Shading Modes](#) i.e. *Solid* or *Wireframe* mode.

Transparent

Creates a transparent background. This is useful for doing overlays like rendering out Grease Pencil films via the Sequencer.

Sound

Strip Volume

Volume of the audio taken from the chosen scene.

Limitations

Scene strips do not render individual [Render Passes](#); only the *Combined* render pass will be used.

Clip Strip

Clip can be modified within the [Movie Clip Editor](#).

Options

Reference

Panel:: *Sidebar region ▶ Strip ▶ Movie Clip*

Movie Clip

Used to select the movie clip. For controls see [Data-Block Menu](#).

Use: 2D Stabilized Clip

Use the 2D stabilized version of the clip.

Use: Undistorted Clip

Use the undistorted version of the clip.

Below the properties, shows the frame range of the movie clip (before strip adjustments).

Mask Strip

The Mask strip generates a mask image from the selected mask data-block generated in the [Movie Clip Editor](#). This works similar to the [Mask Node](#) but without the options available for finer control. The mask image is always generated at the render resolution, scaling along with different proxy levels.

Options

Mask

[Data-block menu](#) to select a mask.

Movie Strip

To add a movie (with or without audio) select a movie file(s) in the File Browser e.g. in the Audio-Video Interleaved format (*.avi file).

Note

Clips can be Huge

A three minute QuickTime .mov file can be 140MB. Loading it, even over a high-speed LAN can take some time. Do not assume your computer or Blender has locked up if nothing happens for awhile.

Tip

Movie strips can display thumbnails in the Sequencer overlaid on their strips by enabling the [Thumbnails](#) overlay.

Add Movie Strip

Reference

Menu:: *Add ▶ Movie*

Relative Path

Store the location of the image file relative to the blend-file.

Start Frame

The [Start Frame](#) to place the left handle of the strip.

Channel

The [Channel](#) to place the strip.

Replace Selection

Replaces the currently selected strips with the new strip.

Fit Method

Determines how images with an aspect ratio different than the scene's [Resolution](#) are scaled to fit inside the render area.

Scale to Fit:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fit exactly within the project's [Resolution](#) while maintaining the original aspect ratio.
This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to span the project's [Resolution](#) while maintaining the original aspect ratio.
This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fill the project's [Resolution](#). Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.
This may mean that the original image becomes distorted to fit the content inside the rendered area.

Set View Transform

Automatically sets an appropriate [View Transform](#) based on the [Color Space](#) of the imported media. In most cases, the *Standard* should be used; using the wrong transform could result in inaccurate colors or degraded rendering performance.

Adjust Playback Rate

Automatically adjusts the video's speed to playback at the original speed regardless of the scene's framerate.

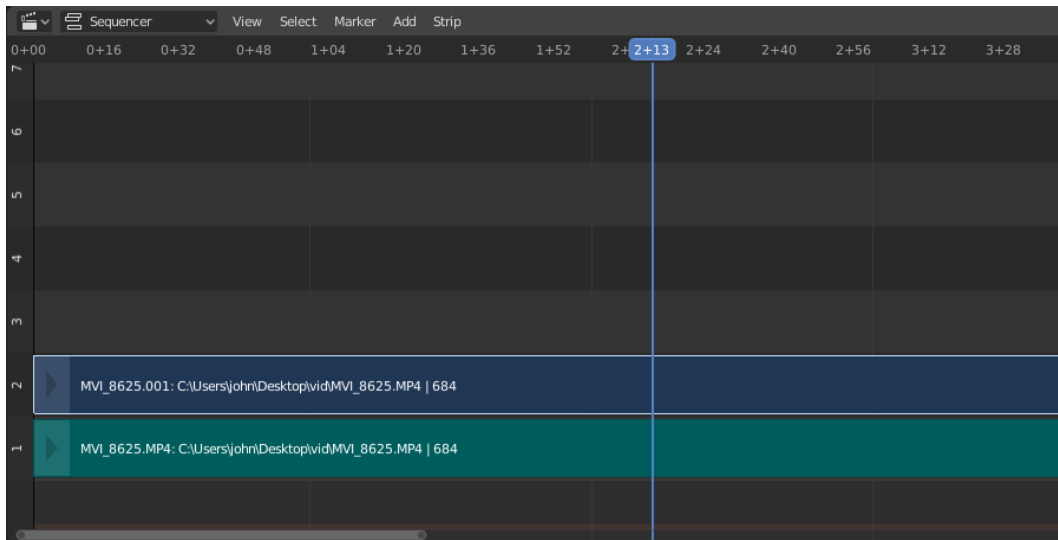
Sound

Add a [Sound Strip](#) that contains the movie's audio track.

Set Scene Frame Rate

Sets the [Scene Frame Rate](#) to the frame rate encoded in the movie file.

Example

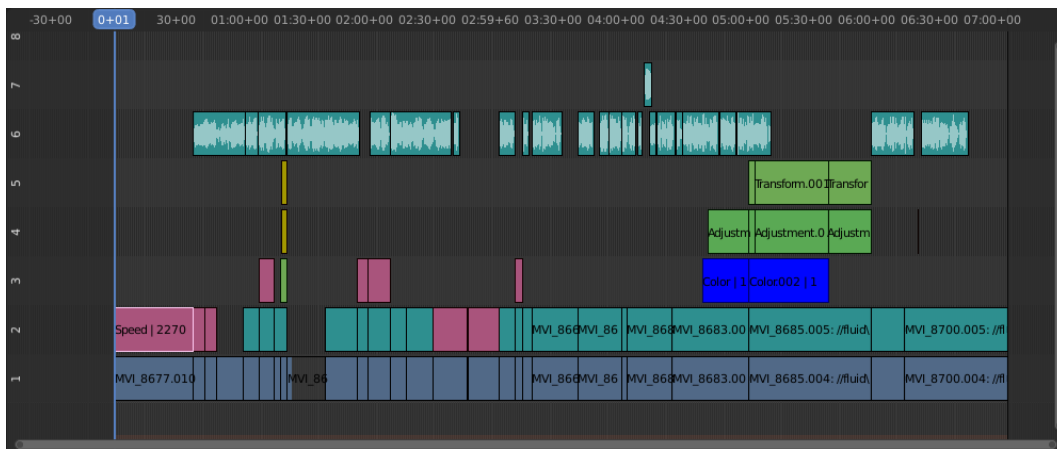


Imported Movie strip with audio track underneath.

In the strip itself, you can see strip name, path to source file, and strip length.

Sound Strip

As well as images and movies the Video Sequencer can also edit audio tracks. You can add Waveform Audio format WAV, mp3 and other audio formats files from your drive, or from sound encoded within a movie, and mix them using an F-Curve as a volume control.



Example of sound editing.

Working with Audio Tracks

A Sound strip is just like any other strip in the Video Sequencer. You can select and move it, adjust its starting offset using LMB over the strip handles, and K cut it into pieces. A useful example is cutting out the “um’s” and dead voice time.

You can have as many Sound strips as you wish and the result will be the mixing of all of them. You can give each strip its own name and volume via the Sidebar region.

Overlapping strips are automatically mixed down during the rendering process. For example, you can have the announcer on channel 5, background music on channel 6, and Foley sound effects on channel 7.

See also

In the [Playback Popover](#) menu of the Timeline you will find some options concerning audio playback behavior.

Waveform

The waveform of the audio is shown depending on two options:

Overlay

The Sequencer Overlay menu has options to show all strip wave-forms, none of them, or to use the per-strip option described below.

Strip

Each strip has an option *Display Waveform*. It is only visible when the above overlay option is set to *Use Strip Option*.

Clipping audio, i.e. values over 100% amplitude, will be shown in red in the waveform.

More strip options are documented in [Sound Sidebar Panel](#).

Animating Audio Track Properties

To animate Sound strips simply hit I over any of its values. Examples of animating an audio strip are to fade in/out background music or to adjust volume levels. Layered/crossed Sound strips are added together; the lower channel does not override and cut out higher channels (unlike image and video strips). This makes Blender an audio mixer. By adding audio tracks and using the curves to adjust each tracks sound level, you have an automated dynamic multi-track audio mixer!

See also

Sounds can be cross-faded by adding a [Sound Crossfade](#) effect.

Output

There are two ways to render out your audio. You can either have it encoded with a video file or in its own audio file. Read more on how to select a proper [audio format](#) and how to start [rendering](#).

Add Sound Strip

Reference

Menu:: *Add ▶ Sound*

Relative Path

Store the location of the image file relative to the blend-file.

Start Frame

The [Start Frame](#) to place the left handle of the strip.

Channel

The [Channel](#) to place the strip.

Replace Selection

Replaces the currently selected strips with the new strip.

Cache

Cache the sound in memory, enables [Caching](#) in the Source properties.

Mono

Merge all sound channels into one channel, enables [Mono](#) in the Sound properties.

Image/Sequence Strip

Tip

Image strips can display thumbnails in the Sequencer overlaid on their strips by enabling the [Thumbnails](#) overlay.

Single Image

When you add a single still image (*.jpg, *.png, etc.), Blender creates a 25 frames long strip which will show this image along the strips range.

Image Sequence

In the case of (numbered) image sequences (e.g. *-0001.jpg, *-0002.jpg, *-0003.jpg, etc, of any image format), you have a choice:

Range

Navigate into the directory and LMB click and drag over a range of names to highlight multiple files. You can page down and continue Shift-LMB click-dragging to add more to the selection.

Batch

Shift-LMB click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (jpg, png, exr, etc.).

All

Press A to select/deselect all files in the directory.

Tip

Dealing with Different Sizes

Dealing with different sized images and different sized outputs is tricky. If you have a mismatch between the size of the input image and the render output size, the Video Sequencer will try to auto-scale the image to fit it entirely in the output. This may result in clipping. If you do not want that, use *Crop* and/or *Offset* in the Input panel to move and select a region of the image within the output. When you use *Crop* or *Offset*, the auto-scaling will be disabled and you can manually re-scale by adding the Transform effect.

Add Image Strip

Reference

Menu:: *Add ▶ Image/Sequence*

Relative Path

Store the location of the image file relative to the blend-file.

Start Frame

The [Start Frame](#) to place the left handle of the strip.

End Frame

The end frame to place the right handle of the strip.

Tip

Subtract the *Start Frame* from the *End Frame* to get the strip's duration.

Channel

The [Channel](#) to place the strip.

Replace Selection

Previously selected strips will be deselected. Only added strips will be selected.

Fit Method

Determines how images with an aspect ratio different than the scene's [Resolution](#) are scaled to fit inside the render area.

Scale to Fit:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fit exactly within the project's [Resolution](#) while maintaining the original aspect ratio.
This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to span the project's [Resolution](#) while maintaining the original aspect ratio.
This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill:: Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fill the project's [Resolution](#). Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.
This may mean that the original image becomes distorted to fit the content inside the rendered area.

Set View Transform

Automatically sets an appropriate [View Transform](#) based on the [Color Space](#) of the imported media. In most cases, the *Standard* should be used; using the wrong transform could result in inaccurate colors or degraded rendering performance.

Use Placeholders

Image sequences can use placeholder files. This works by enabling *Use Placeholders* checkbox when adding an image strip. The option detects

the frame range of opened images using Blender's frame naming scheme (filename + frame number + .extension) and makes an image sequence with all files in between even if they are missing. This allows you to render an image sequence with a few frames missing and still the image strip will have the correct range to account for the missing frames displayed as black.

When the missing frames are rendered or placed in the same folder, you can [refresh](#) the Sequencer and get the missing frames in the strip. The option is also available when using the *Change Data/File* operator and allows you to add more images to the range.

Color Strip

This effect generates solid color frames. By default, when it is created, the Color strip is 25 frames long, but you can extend it by selecting and moving one of the ends. Use this strip crossed with your main movie to provide a fade-in or fade-out.

Options

Color

Click on the color field in the Effect panel in the Sidebar region, to pick a different color.

Text Strip

The Text strip allows you to directly display text in the Sequence editor. The strip will display the text inserted in its text field on the final sequence.

Tip

All Text strips in a video sequence can be [exported](#) as a [SubRip](#) file. This is useful when using Text strips as subtitles.

Options

Text

The actual text displayed.

Wrap Width

Wraps the text by the percentage of the frame width, setting this to zero disables word wrapping.

Style

Font

[Data-Block Menu](#) to choose which font-file is used to render the text.

Bold

Use a bold font face with a strong/thick visual appearance.

Italics

Use an italicized font face with a slanted visual appearance.

Size

Size of the text.

Color

The text color.

Shadow

Creates a shadow of the specified color under the text.

Shadow Angle

Defines the position of the shadow as an angle, 0° being to the right and 90° being below.

Shadow Offset

Amount to shift the shadow compared to the normal text.

Shadow Blur

Amount to blur the shadow.

Outline

Creates a line with the defined color enclosing the shape of the text.

Outline Width

The thickness of the outline.

Box

Creates a background for the text to improve the readability and clarity of text in some situations. The color and opacity of the box can be adjusted using the color selector.

Box Margin

The distance the box boundaries extends from the boundaries of the font glyphs. The distance is measured as a factor of the image's width.

Layout

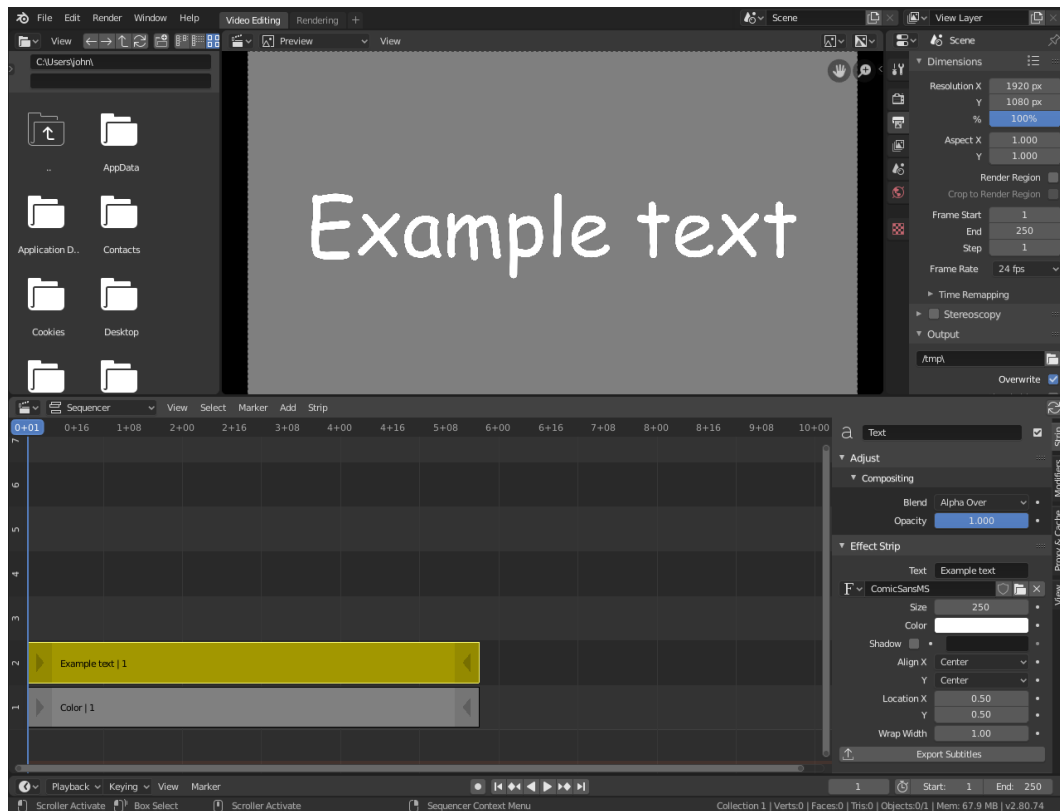
Location X, Y

Positions the text on the X, Y axis.

Anchor X, Y

Horizontal (X) or vertical (Y) anchor point of the text relative to the location.

Example



Text effect.

Adjustment Layer Strip

The Adjustment Layer strip works like a regular input file strip except for the fact, that it considers all strips below it as its input.

Real-world use cases, you want to add some last finishing color correction on top of parts of your final sequence, timeline without messing with meta strips around. Just add an adjustment layer on top and activate the color balance.

Or you can stack a primary color correction and several secondary color corrections on top of each other (probably using the new mask input for area selection).

Options

This strip has no options.

Effect Strips

- [Add Strip](#)
- [Subtract Strip](#)
- [Multiply Strip](#)
- [Alpha Over, Under & Over Drop Strips](#)
- [Color Mix Strip](#)
- [Multicam Selector Strip](#)
- [Transform Strip](#)
- [Speed Control Strip](#)
- [Glow Strip](#)
- [Gaussian Blur Strip](#)

Add Strip

The Add effect strip adds the colors of two strips together. Use this effect with a base image strip, and a modifier strip. The modifier strip is either a solid color or a black-and-white mask, or another image entirely.

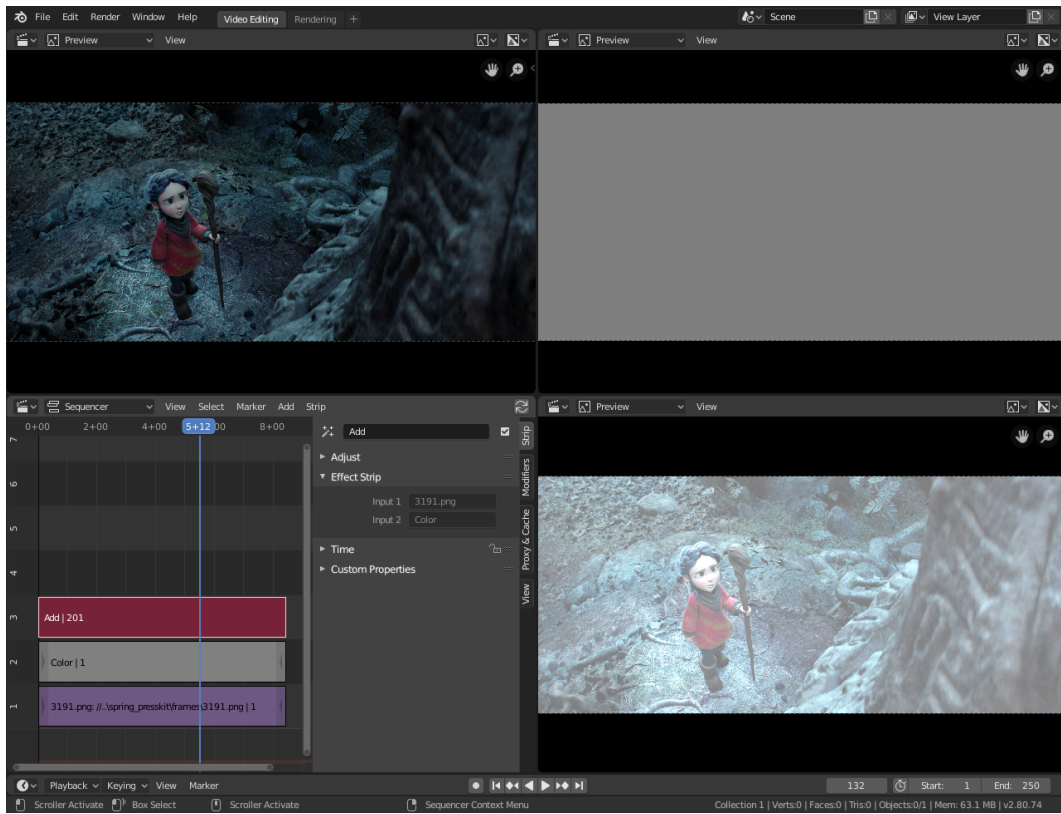
You can use this effect to increase the brightness of an image, or if you use a BW mask, selectively increase the brightness of certain areas of the image. The Mix node, in Add mode, does exactly the same thing as the Add SFX strip here, and is controlled the same way by feeding the Factor input.

The [example](#) shows what happens when you add gray to an image. The image gets bright because we are adding gray RGB(0.5, 0.5, 0.5) to say, a blue color RGB(0.1, 0.1, 0.5) resulting in RGB(0.6, 0.6, 1.0) which retains the original hue (relationship between the colors) but is much brighter (has a higher value). When applied to the whole image like this, it seems to flash.

Options

This strip has no options.

Example



Add Effect.

Subtract Strip

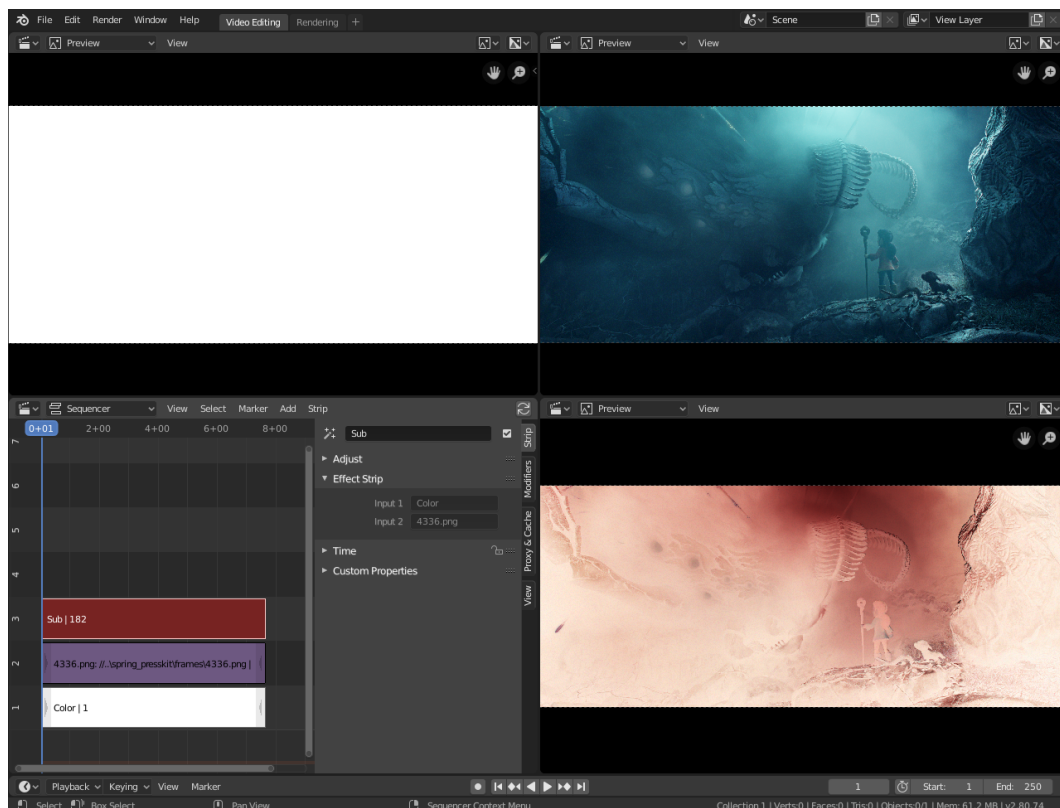
This effect takes away one strip's color from the second.

Make a negative of an image using this effect, or switch the order of the strips and just darken the strip. Subtracting a hue of blue from a white image will make it yellow, since red and green make yellow.

Options

This strip has no options.

Example



Subtract Effect.

Multiply Strip

The *Multiply* effect multiplies two colors. Blender uses values between (0.0 to 1.0) for the colors. This operation does not have to be normalized, the multiplication of two terms between (0.0 to 1.0) always gives a result between (0.0 to 1.0).

(With the “traditional” representation of three bytes, like RGB(124, 255, 56), the multiplications give far too high results, like RGB(7316, 46410, 1848), that have to be normalized (brought back) by dividing them by 256 to fit in the range of (0 to 255)...)

This effect has two main usages:

With a Mask

A mask is a black-and-white picture which, after multiplication with a “normal” image, only show this one in the white areas of the mask (everything else is black).

The opening title sequence to James Bond movies, where the camera is facing down the barrel of a gun at James, is a good example of this effect.

With Uniform Colors

Multiplying a color with a “normal” image allows you to soften some hues of this one (and so – symmetrically – to enhance the others).

For example, if you have a brown pixel RGB(0.50, 0.29, 0.05), and you multiply it with a cyan filter (uniform color RGB(0.0, 1.0, 1.0)), you will get a color RGB(0.0, 0.29, 0.05). Visually, the result is to zero the reds and bring up (by “symmetry” – the real values remain unchanged!) the blues and greens. Physically, it is the same effect as shining a cyan light onto a chocolate bar. Emotionally, vegetation becomes more lush, water becomes more Caribbean and inviting, skies become friendlier.

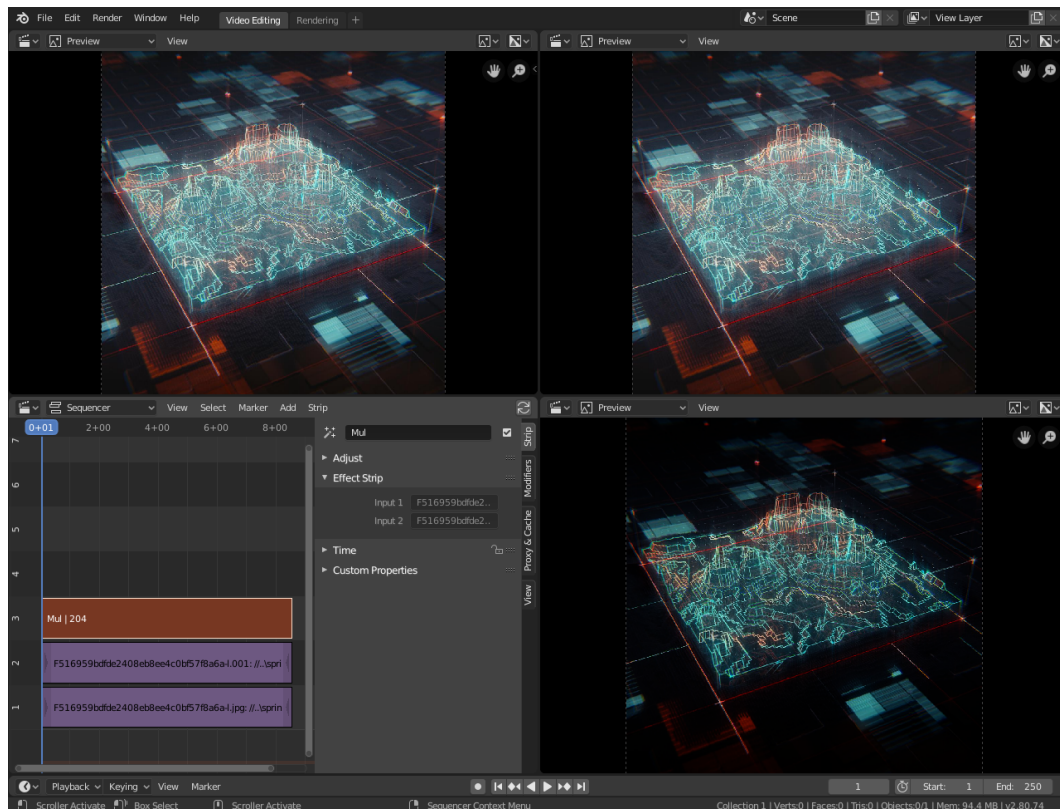
Note

This effect reduces the global luminosity of the picture (the result will always be smaller than the smallest operand). If one of the images is all white, the result is the other picture; if one of the images is all black, the result is all black!

Options

This strip has no options.

Example



Multiply Effect.

Alpha Over, Under & Over Drop Strips

Using the alpha (transparency channel), this effect composites a result based on transparent areas of the dominant image. If you use a Scene strip, the areas of the image where there is not anything solid are transparent; they have an alpha value of 0. If you use a Movie strip, that movie has an alpha value of 1 (completely opaque).

So, you can use the *Alpha Over / Alpha Under* effect to composite the CGI Scene on top of your movie. The result is your model doing whatever as if it was part of the movie. The *Adjust ▶ Compositing ▶ Opacity* controls how much the foreground is mixed over the background, fading in the foreground on top of the background. The colors of transparent foreground image areas are ignored and do not change the color of the background.

Alpha Over

With *Alpha Over*, the strips are layered up in the order selected; the first strip selected is the background, and the second one goes *over* the first one selected. The *Opacity* controls the transparency of the *foreground*, i.e. *Opacity* of 0.0; will only show the background, and an *Opacity* of 1.0 will completely override the background with the foreground (except in the transparent areas of this one, of course!)

Warning

By clicking the *Premultiply Alpha* button in the Sidebar of the foreground strip, the alpha values of the two strips are not multiplied or added together. Use this effect when adding a foreground strip that has a variable alpha channel (some opaque areas, some transparent, some in between) over a strip that has a flat opaque (alpha=1.0 or greater) channel. If you

notice a glow around your foreground objects, or strange transparent areas of your foreground object when using *Alpha Over*, enable *Premultiply*.

Example



Alpha Over Effect.

Alpha Under

With *Alpha Under*, this is the contrary: The first strip selected is the foreground, and the second one, the background. Moreover, the *Opacity* controls the transparency of the *background*, i.e. an *Opacity* of 0.0; will only show the foreground (the background is completely transparent), and an *Opacity* of 1.0 will give the same results as with *Alpha Over*.

Over Drop

Over Drop is between the two others: as with *Alpha Under*, the first selected strip will be the foreground, but as with *Alpha Over*, the *Opacity* controls the transparency of this foreground.

The *Over Drop* effect is much like the Cross, but puts preference to the top or second image, giving more of a gradual overlay effect than a blend like the Cross does. Of course, all of the Alpha effects respect the alpha (transparency) channel, whereas Cross does not.

The degree of Alpha applied, and thus color mixing, can be controlled by an F-Curve. Creating a Sine wave could have the effect of the foreground fading in and out.

Color Mix Strip

The *Color Mix* effect strip mixes two strips by working on the individual and corresponding pixels of the two input strips.

This effect can do the exact same operation as the Add, Subtract, or Multiply effect strips but also other color blending modes.

Options

Blend

The Blend modes can be selected in the select menu. See [Color Blend Modes](#) for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Opacity

The amount of the blend of the second image gets composed onto the first.

Multicam Selector Strip

The Multicam Selector strip is used for multi-camera editing. Multi-camera editing is when a scene is recorded using multiple cameras from different angles and then edited together afterwards.

Workflow

The process of multi camera editing can be rather easy in the Video Sequencer if properly setup. The following guide shows the basic steps to setup a basic multi camera editing workflow.

1. First you are going to want to add in each of your video strips.
2. Next, you will want to sync all your cameras by either using [Audio Waveforms](#) or by the movement of objects.

Tip

To make syncing strips easier you can group cameras, their audio, and their effects together using [Meta Strips](#).

3. Split the editor into many [Previews](#), one for each input track. Then change the [Display Channel](#) of each of the previews to the channel number of the input track.
4. Add a Multicam Selector strip *above* all the video channel tracks.

After completing these steps you should get something similar to the following image:



Multi-camera editing setup.

5. Now select the Multicam strip, if you take a look at the strip options (in the Sidebar), you will notice, that Multicam is a rather simple effect strip: It just takes a selected channel as its input. That is all. The magic comes with the convenient keyboard layout.
6. When you select the Multicam strip, the keys 1 to 9 are mapped to the cut buttons. So, select the Multicam strip and start playback and press the keys for the correct input while watching the individual cameras.
7. You will end up with a small Multicam Selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what is coming, watch it again and do a rough cut using the number keys. Then fine-tune the placement by selecting the outer handles of two neighboring Multicam for A/B rolling.

Tip

To improve playback performance enable [Proxies](#).

Options

Source Channel

The channel which the Multicam Selector gets its input from.

Cut To

Cuts the Multicam strip at the current frame and changes the *Source Channel* automatically to the selected channels.

Workflow

1. First you are going to want to add in each of your video strips.
2. Next, you will want to sync all your cameras by either using [Audio Waveforms](#) or by the movement of objects.

Tip

To make syncing strips easier you can group cameras, their audio, and their effects together using [Meta Strips](#).

3. Split the editor into many [Previews](#), one for each input track. Then change the [Display Channel](#) of each of the previews to the channel number of the input track.
4. Add a Multicam Selector strip *above* all the video channel tracks.

After completing these steps you should get something similar to the following image:



Multi-camera editing setup.

5. Now select the Multicam strip, if you take a look at the strip options (in the Sidebar), you will notice, that Multicam is a rather simple effect strip: It just takes a selected channel as its input. That is all. The magic comes with the convenient keyboard layout.
6. When you select the Multicam strip, the keys 1 to 9 are mapped to the cut buttons. So, select the Multicam strip and start playback and press the keys for the correct input while watching the individual cameras.
7. You will end up with a small Multicam Selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what is coming, watch it again and do a rough cut using the number keys. Then fine-tune the placement by selecting the outer handles of two neighboring Multicam for A/B rolling.

Tip

To improve playback performance enable [Proxies](#).

Transform Strip

Transform is a swiss-army knife of image manipulation. It moves, rotates, and scales the images within a strip.

Options

Interpolation

None::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

Translation Unit

Control whether the input values are in *Percent* or *Pixels*.

Position

Moves the input along the X and Y axis.

Uniform Scale

Scale the input evenly along the X and Y axis.

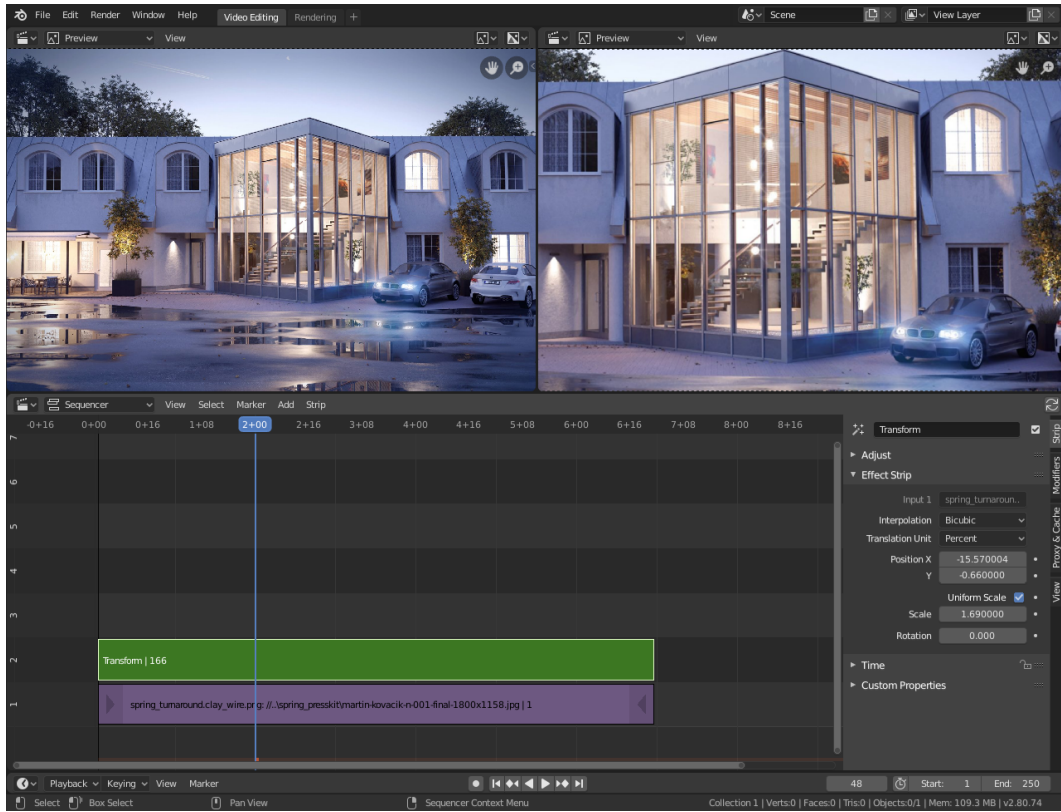
Scale

Scale the image on the X and Y axis.

Rotation

Rotates the input two-dimensionally along the Z axis.

Example



Transform Effect.

Speed Control Strip

Speed Control time-warps the strip, making it play faster or slower than it normally would. Playing faster means that some frames are skipped, and the strip will run out of frames before the end frame. When the strip runs out of frames to display, it will just keep repeating the last one action that will appear to be frozen. To avoid this, position the next strip under the original at a point where you want the motion to continue.

Options

Speed Control

The method used to adjust the speed of the strip.

Stretch:: Automatically calculates the speed effect based on the length of the input strip. If you scale a strip to 1/2 the original size the sequence will play back at 2 times the speed.

Multiply:: Multiplies the current speed of the sequence by a *Multiply Factor*. Thus a value of 0.5 will make the sequence half as fast while 2 would make the sequence twice as fast. Negative values will reverse the input while also adjusting the speed, so a value of negative two will play in reverse and twice as fast as normal.

Note

You will have to manually re-adjust the length of the strip accordingly.

Frame Number:: Specifies a frame to remap the current frame to, for example, setting the *Frame Number* value to 50 displays the 50th frame. This can then be manually [keyframed](#) to recreate the animation.

Length:: Maps the frame range on a percentage scale. For example, using this and a value of 50% will select the frame halfway through the sequence.

Interpolation

Crossfades between frames to reduce screen tearing when the speed is slower than the original frame rate.

Examples

Creating a Slow-Motion Effect

Suppose you want to slow your strip down. You need to affect the speed of the video clip without affecting the overall frame rate. Select the clip and *Add ▶ Effect ▶ Speed Control* effect strip.

Choose the *Multiply* option in the Effect Strip panel in the Sidebar. Set the Multiply Factor to be the factor by which you want to adjust the speed. To cut the displayed speed by 50%, enter 0.5. Now, a 275-frame clip will play at half speed, and thus display only the first 137 frames.

If you want the remaining frames to show in slow motion after the first set is displayed, double the Length of the source strip (since effects strip bounds are controlled by their source strips). If you are using a speed factor other than 0.5 then use the formula:

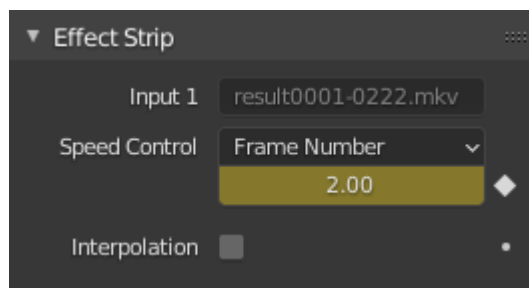
$$\text{new\_length} = \text{real\_length} / \text{speed\_factor}$$

That is it, set your render to animate (in this example) all 550 frames.

Keyframing the Speed Control

To get even finer control over your clip timing, you can use curves! While it is possible to keyframe the Multiply factor, usually you want to keyframe the Frame number directly.

Choose the *Frame Number* option. You now have a Frame number field which you can keyframe. If you want the strip to animate **at all** you will have to insert some keyframes, otherwise it will look like a still. In most cases you will want to use the Graph editor view to set the curve interpolation to Linear since the default Bézier will rarely be what you want.



Keyframing the Frame number.

Tip

If you choose to keyframe the Speed factor instead, remember to [Refresh All](#) or the changes will not take effect.

Changing Video Frame Rates

You can use the speed control to change the frame rate in frames per second (fps) of a video. If you are rendering your video to a sequence set, you can effectively increase or decrease the number of individual image files created, by using a Multiply value less than or greater than one, respectively.

For example, if you captured a five-minute video at 30 fps and wanted to transfer that to film, which runs at 24 fps, you would enter a Multiply Factor of $30/24$, or 1.25 (and enable *Interpolation* frame blending to create a film blur effect). Instead of producing $5 \times 60 \times 30 = 9000$ frames, Blender would produce $9000 / 1.25 = 7200 = 5 \times 60 \times 24$ frames. In this case, you set a *start* = 1 and *end* = 7200, set your Format output to jpeg 30fps, and image files 0001.jpg through 7200.jpg would be rendered out, but those images cover the entire 9000 frames. The image file 7200.jpg is the same at frame 9000. When you read those images back into your film blend-file at 24 fps, the strip will last exactly 5 minutes.

Glow Strip

This effect makes parts of an image glow brighter by working on the luminance channel of an image. The *Glow* is the superposition of the base image and a modified version, where bright areas are blurred.

To “animate” the glow effect, mix it with the base image using the Gamma Cross effect, crossing from the base image to the glowing one.

Options

Threshold

Areas brighter than the *Threshold* are blurred.

Clamp

The maximum luminosity that is added.

Boost Factor

Multiplier of the brightness.

Blur Distance

The size of the blur.

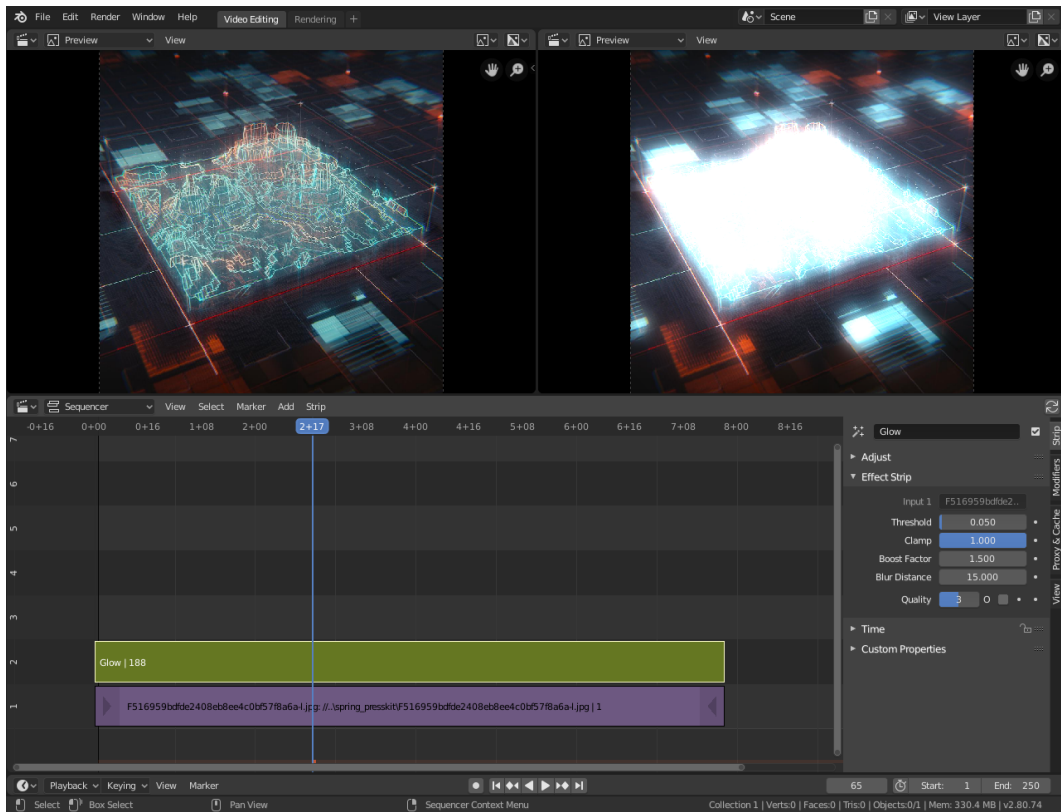
Quality

Improves the quality of the glow by giving smoother results but will be slower.

Only Boost

This checkbox allows you to only show/use the “modified” version of the image, without the base one.

Example



Glow effect.

Gaussian Blur Strip

The Gaussian Blur strip is used to blur the input strip in a defined direction. This can be used to blur a background or to blur a transition strip.

Options

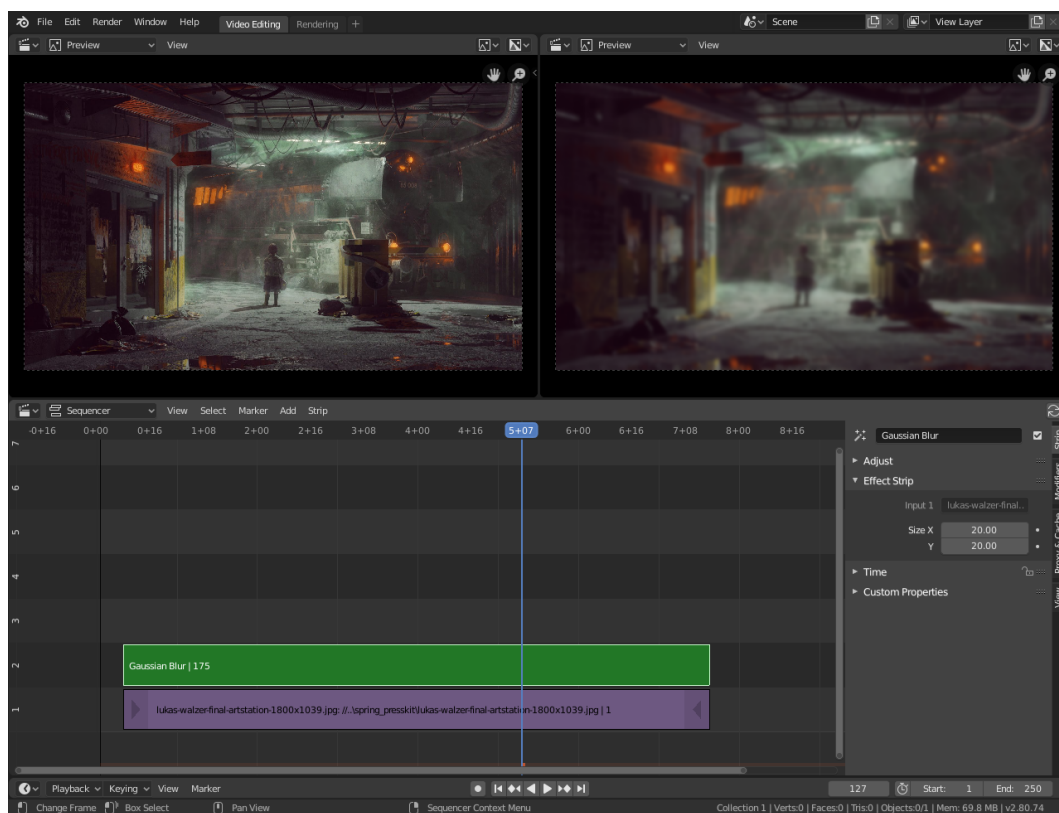
Size X

Distance of the blur effect on the X axis.

Size Y

Distance of the blur effect on the Y axis.

Example



Gaussian Blur Effect.

Transitions

- [Sound Crossfade](#)
- [Cross Strip](#)
- [Gamma Cross Strip](#)
- [Wipe Strip](#)

Sound Crossfade

The *Sound Crossfade* transition works by animating the *Volume* of two overlapping Sound strips to evenly fade between them. Because this simply animates a value it does not create a strip like other effects or transitions.

Cross Strip

The *Cross* transition fades from one strip to another, also known as a crossfade. Strips can be overlapping or have a gap between them, however, when strips contain a gap the last and first frame of each strip is extended which can cause a pause if any of the strips are a sequence.

Options

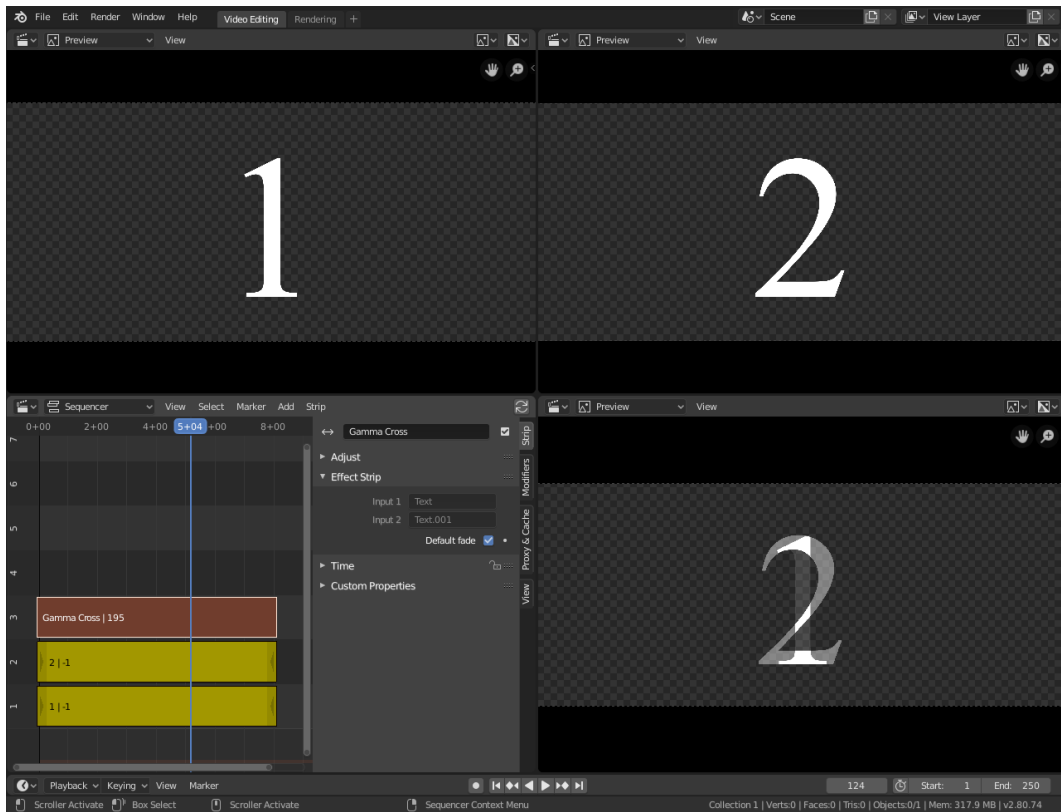
Default Fade

Automatically calculates a linear fade over the length of the strip.

Effect Fader

Allows you to manually [keyframe](#) a custom fade. This can be used with different [easings](#) to fine-tune the fade in/out.

Example



Cross Effect.

Gamma Cross Strip

The Gamma Cross transition is similar to the [Cross Strip](#) transition, however, the Gamma Cross strip transition uses color correction while transitioning between the two strips, resulting in a smoother transition that is easier on the eyes.

Options

Default Fade

Automatically calculates a linear fade over the length of the strip.

Effect Fader

Allows you to manually [keyframe](#) a custom fade. This can be used with different [easings](#) to fine-tune the fade in/out.

Wipe Strip

The *Wipe* transition strip can be used to transition from one strip to the next. The wipe will have no effect if created from a single strip instead of two strips. The duration of the wipe is the intersection of the two source strips and cannot be adjusted. To adjust the start and end of the wipe you must adjust the temporal bounds of the source strips in a way that alters their intersection.

Options

Transition

The type of transition used.

Single

Reveals the next strip by uncovering it in a straight line moving across the image.

Double

Similar to *Single*, but uses two lines either starting from the middle of the image or the outside. Like the blink of an eye.

Iris

Reveals the next strip through an expanding (or contracting) circle. Like the aperture of a camera or pupil of an eye. You can blur the transition, so it looks like ink bleeding through a paper.

Clock

Like the hands of an analog clock, it sweeps clockwise or (if Wipe In is enabled) counterclockwise from the 9:00 position. As it sweeps, it reveals the next strip.

Direction

Controls whether to fade *In* or *Out*.

Blur Width

The width of the blur used to blur the transition.

Angle

Controls the angle of the line for *Single* and *Double* transition types.

Default Fade

Automatically calculates a linear fade over the length of the strip.

Effect Fader

Allows you to manually [keyframe](#) a custom fade. This can be used with different [easings](#) to fine-tune the fade in/out.

Example



Wipe Effect.

Selecting

The active sequence strip is displayed with a light outline. The *entire* strip could be selected by clicking LMB in the middle of the strip.

Select Menu

The Select menu lets you select strips in different ways.

All A

Selects all the strips in the timeline.

None Alt-A

Deselects all the strips in the timeline.

Invert Ctrl-I

Inverts the current selection.

Box Select B

See [Box Select](#).

Box Select (Include Handles) Ctrl-B

Works like *Box Select*, but also selects any strip handles inside the box.

If a strip has only one handle selected, dragging it will change the strip's length. (If both handles are selected, the complete strip moves instead.)

Select Grouped Shift-G

Select strips that are similar to the active strip. By default, unsimilar strips are deselected, but this can be changed in the [Adjust Last Operation](#) region.

Type

Select strips that have the same specific type as the active strip. For example, if the active strip is a Movie strip, this selects all Movie strips.

Global Type

Select strips that have the same general type (graphics or audio) as the active strip.

Effect Type

If the active strip is an effect strip, selects all effect strips.
Otherwise, selects all non-effect strips. (Despite the name, this operator does not check the effect type.)

Data

Select strips that use the same source (file, scene, movie clip or mask) as the active strip.

Effect

Find the effect types that are applied to the active strip, and select all strips that have any of the same effect types applied to them. For example, if the active strip has a Gaussian Blur effect on it, this will select all other strips that are also blurred.

Effect/Linked

Select strips that are on a lower channel than a selected strip and overlap it in time; then, effect strips linked to the selected content strips; and finally, content strips linked to the selected effect strips.

Overlap

Select strips that partially or completely overlap the active strip in time.

Select Linked

All Ctrl-L / Less Ctrl-NumpadMinus / More Ctrl-NumpadPlus
Add/remove neighboring strips to/from the selection.

Side of Frame

Left/Right [/]

Select the strips that lie completely to the left or right of the current frame.

Current

Select the strips that intersect the current frame.

Handle

Both, Left, Right

Select the left, right, or both handles of the selected strips.

Both/Left/Right Neighbor

Select the handle of the neighboring strip to the left, right, or on both sides of the selected strips.

Channel

Select all the strips that are in the same channels as the currently selected strips.

Editing

Transform

Options

Overlap Mode

Overlap Mode defines the result of transforming a strip so that it overlaps another strip.

Shuffle

The overlapping strip will be moved to the nearest free space so that it does not overlap.

Overwrite

The overlapped strip will be overwritten, trimmed or split by the overlapping strip.

Expand

All strips on the right side of (each) transformed will be shifted forward to accommodate the overlapping strip.

Snapping

The icon toggles snapping; you can also do this temporarily by holding `Ctrl` after starting to drag a strip.

The drop-down arrow offers the following options:

Snap to

Current Frame

Snaps the transformed selection to the Playhead.

Hold Offset

Snap the transformed selection to the [Hold Offset](#).

Markers

Snap the transformed selection to [Markers](#).

Ignore

Muted Strips

Muted Strips are not considered as snap targets.

Sound Strips

Sound Strips are not considered as snap targets.

Current Frame

Snap to Strips

Snap the Playhead to all strips.

Move

Reference

Menu:: *Strip ▶ Transform ▶ Move*

Shortcut:: G

Pressing G moves all the selected strip(s). Move your mouse horizontally (left/right) to change the strip's position in time. Move vertically (up/down) to change channels.

Holding down Ctrl while dragging enables or disables snapping.

You can also lock the direction to time with X or to change the strip's channel with Y.

It is possible to move strips using mouse by dragging them while holding LMB. Currently it is possible to move only one strip by dragging.

Start Frame Offset

The *Start Frame Offset* for that strip can be selected by clicking LMB on the left handle of the strip; holding it down (or pressing G and then moving the mouse left/right) changes the start frame within the strip by the number of frames you move it. The frame number label under the strip displays the start frame of the strip.

- If you have a 20-image sequence strip, and drag the left handle to the right by 10 frames, the strip will start at image 11 (images 1 to 10 will be skipped). Use this to clip off a roll-up or undesired lead-in.
- Dragging the left handle left will create a lead-in (copies) of the first frame for as many frames as you drag it. Use this when you want some frames for a transition at the start of the clip.

End Frame

The *End Frame* of the strip could be selected by clicking LMB on the right handle of the strip; holding it down (or pressing G) and then moving the mouse changes the ending frame within the strip. The frame number label over the strip displays the end frame of the strip.

- Dragging the right handle to the left shortens the clip; any original images at the tail are ignored. Use this to quickly clip off a roll-down.
- Dragging the right handle to the right extends the clip. For movies and images sequences, more of the animation is used until exhausted. Extending a clip beyond its length will render as a copy of the last image. Use this for transitions out of this clip.

Note

Multiple selection

You can select several (handles of) strips by Shift-LMB clicking: when you press G, everything that is selected will move with your mouse – this means that, for example, you can at the same time move a strip, shorten two others, and extend a forth one.

Move/Extend from Current Frame

Reference

Menu:: *Strip ▶ Transform ▶ Move/Extend from Current Frame*

Shortcut:: E

With a number of strips selected, pressing E lets you interactively extend the strips. This is similar to moving but is useful for extending (or shortening) time around the current frame.

All selected strip handles to the “mouse side” of the current frame indicator will transform together, so you can change the duration of strips at the current frame.

Hint

Extend is a convenient way to adjust the time of rough edits such as an “animatic” (sequential storyboards). Where it’s possible to select everything and adjust the length of strips around the current frame. This can be especially useful when adding in audio or other elements that could cause the timing to need adjustment.

When performing this operation you may want to enable *Markers ▶ Sync Markers* so markers are updated too.

This is simply a convenience operation, instead of manually selecting strips on one side of the current frame, as well as handles on one side of overlapping strips. Then selecting and transforming markers as well. This avoids the manual process, so re-timing can be accessed quickly.

Slip Strip Contents

Reference

Menu:: *Strip ▶ Transform ▶ Slip Strip Contents*
Shortcut:: S

The Slip tool allows you to change the position of the contents of a strip without moving the strip itself.

Snap Strips to the Current Frame

Reference

Menu:: *Strip ▶ Transform ▶ Snap Strips to the Current Frame*
Shortcut:: Shift-S

Moves the strip or control point to the current frame.

Clear Strips Offset

Reference

Menu:: *Strip ▶ Transform ▶ Clear Strips Offset*
Shortcut:: Alt-0

To reset the (soft) start/end frame handles.

Swap Strips

Reference

Menu:: *Strip ▶ Transform ▶ Swap Strips*

Left Alt-Left

Swaps the active strip with the strip to the left.

Right Alt-Right

Swaps the active strip with the strip to the right.

Remove Gaps

Reference

Menu:: *Strip ▶ Transform ▶ Insert Gaps*

Shortcut:: Backspace

Remove blank frames between the current frame and the first strip to the left, independent of selection or locked state of strips.

All Gaps

Remove gaps to the right of the strip along with the left.

Insert Gaps

Reference

Menu:: *Strip ▶ Transform ▶ Insert Gaps*

Shortcut:: Equals

Insert blank frames between the current frame and the first strips to the right, independent of selection or locked state of strips.

Image Transform

Scale to Fit

Reference

Menu:: *Strip ▶ Image Transform ▶ Scale to Fit*

Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fit exactly within the project's [Resolution](#) while maintaining the original aspect

ratio.

This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill

Reference

Menu:: *Strip ▶ Image Transform ▶ Scale to Fill*

Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to span the project's [Resolution](#) while maintaining the original aspect ratio.

This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill

Reference

Menu:: *Strip ▶ Image Transform ▶ Stretch to Fill*

Adjusts the strips [Scale Transforms](#) so the visual contents of the strip to fill the project's [Resolution](#). Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.

This may mean that the original image becomes distorted to fit the content inside the rendered area.

Clear Position

Reference

Menu:: *Strip ▶ Image Transform ▶ Clear Position*

Resets the strips [Position Transforms](#) to a value of zero.

Clear Scale

Reference

Menu:: *Strip ▶ Image Transform ▶ Clear Scale*

Resets the strips [Scale Transforms](#) to a value of one.

Clear Rotation

Reference

Menu:: *Strip ▶ Image Transform ▶ Clear Rotation*

Resets the strips [Rotation Transform](#) to a value of zero.

Clear All

Reference

Menu:: *Strip ▶ Image Transform ▶ Clear All*

Resets the strips position, scale, and rotation [Transforms](#) to their default values.

Split

Reference

Menu:: *Strip ▶ Split*

Shortcut:: K

This splits the selected strip in two at the current frame. This will result in two strips which use the same source, fitting the original strip's timing and length.

Hint

This can be thought of as a quick way to duplicate the current strip, adjusting the start/end frames to form two non-overlapping strips showing the same content as before.

Hold Split

Reference

Menu::	<i>Strip ▶ Hold Split</i>
Shortcut::	Shift-K

Like *Split*, it splits a strip in two distinct strips; however you will not be able to drag the endpoints to show the frames past the split of each resulting strip.

Although you can adjust the [Hold Offset](#) number fields in the *Strip Info* panel.

Hint

This can be thought of as a way to simulate splitting the video file in two parts at the cut-point, replacing the current strip with each.

Duplicate Strips

Reference

Menu:: *Strip ▶ Duplicate Strips*
Shortcut:: Shift-D

Duplicate a strip to make an unlinked copy; drag it to a time and channel, and drop it by LMB click.

Delete

Reference

Menu:: *Strip ▶ Delete*
Shortcut:: Delete, X

Delete the selected strip(s).

Update Scene Frame Range

Reference

Menu:: *Strip ▶ Update Scene Frame Range*

For Scene strips only – Updates the strip's [Time](#) properties to match the referenced scene's frame range. This operator should be used when the referenced scene's length is extended or shortened.

Separate Images

Reference

Menu:: *Strip ▶ Separate Images*
Shortcut:: Y

For images sequence only – Converts the strip into multiple strips, one strip for each frame. Useful for slide shows and other cases where you want to

bring in a set on non-continuous images.

Length

You have to specify the duration you want the resulting strips will be.

Movie Strip

Set Render Size

Reference

Menu:: *Strip ▶ Set Render Size*

Sets the render resolution and aspect to match the strip's resolution.

Deinterlace Movies

Reference

Menu:: *Strip ▶ Deinterlace Movies*

Converts interlaced video into progressive video.

Effect Strip

Change Effect Input

Reference

Menu:: *Strip ▶ Effect Strip ▶ Change Effect Type*

Swaps which strips are the input for the effect strip.

Change Effect Type

Reference

Menu:: *Strip ▶ Effect Strip ▶ Change Effect Type*

Switch the effects on a selected Effect strip.

Reassign Inputs

Reference

Menu:: *Strip ▶ Effect Strip ▶ Reassign Inputs*

Shortcut:: R

This tool can be used to assign (reconnect) effect strips in a different way. Select three arbitrary strips and press R. If you don't create a cycle, those will be connected to a new effect chain.

Swap Inputs

Reference

Menu:: *Strip ▶ Effect Strip ▶ Swap Inputs*

Shortcut:: Alt-S

Swaps the first two inputs for the effect strip.

Lock/Unlock

Lock Strips Shift-L

Disables the strip from being transformed.

Unlock Strips Shift-Alt-L

Enables disabled strips allowing them to be transformed.

Mute/Unmute

Mute/Unmute Strips H, Alt-H

Mute or unmute the selected strips.

Mute/Unmute Deselected Strips Shift-H, Ctrl-Alt-H

Mute or unmute all strips but the selected.

Inputs

Reload Strips Alt-R

Reloads the strips from their external saved location.

Reload Strips and Adjust Length Shift-Alt-R

Reloads the strips from their external saved location and re-adjusts the strip duration.

Change Path/Files

Changes the source file contained in a selected strip.

Swap Data

Swaps two sequence strips.

Context Menu

You can activate context menu by clicking RMB in the Sequencer's timeline. In this menu you can quickly access some commonly used tools.

Fades

Reference

Menu:: *Add ► Fades*

This submenu contains tools to add or remove fades to strips. In case of visual strips the tools will animate the opacity or volume in case of audio strips.

Clear Fades

Removes fade animation from selected sequences.

Fade In and Out

Fade selected strips in and out.

Fade In

Fade in selected strips.

Fade Out

Fade out selected strips.

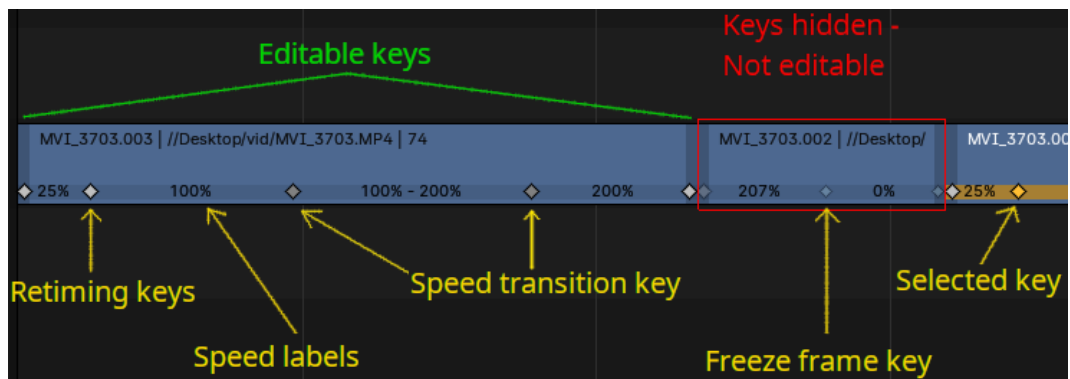
From Current Frame

Fade from the current frame to the end of overlapping sequences.

To Current Frame

Fade from the start of sequences under the Playhead to the current frame.

Retiming



Strips can be sped up or, slowed down by adding and moving retiming keys. Retiming controls can be activated for individual strips, after which keys can be selected and moved.

Note

- Only strip content is retimed, existing animation is not handled by the tool.
- Effect strips can not be retimed.

Hint

To quickly change selected strip speed, press R and enter desired speed.

Selecting Retiming Keys

Retiming keys are always shown on strip as inactive by default. In order to select retiming keys, [Show Retiming Keys](#) must be enabled. This property can also be enabled using [Toggle Retiming Keys](#).

Multiple keys can be selected at once with box selection. Box select will select keys, only if a key is already selected. Otherwise it will select strips only.

Use Ctrl-LMB to select all keys to the right of the selected key.

Moving Retiming Keys

Retiming key can be moved by dragging it with mouse, or by pressing G. The key is mapped to particular frame of strip content, so moving it effectively means moving a frame to a new position and therefore stretching, or contracting time flow.

When a key is moved, this does not affect position of other keys inside of the strip. If strip has more keys inside, multiple keys have to be selected, if only 1 segment has to be retimed. However if there are retiming keys outside of strip boundary, these will be moved along with first or last key in strip in order to preserve existing retiming, that is not visible.

Add Retiming Keys

Reference

Editor:: Video Sequencer

Menu:: *Strip ▶ Retiming ▶ Add Retiming Key*

Retiming key can be added to selected strips from retiming menu or by pressing I and choosing Add Retiming Key option. This adds a key to current frame. This operation will also create keys at strip start and end point, since these keys must be always present.

When keys are selected, strips are deselected, but it is still possible to add new keys. In this case keys will be added to strips where any key is selected.

Add Freeze Frame and Slide

Reference

Editor:: Video Sequencer

Menu:: *Strip ▶ Retiming ▶ Add Freeze Frame and Slide*

Freeze frame is used to stop strip playback at particular frame for any duration. Freeze frame can be added from strip retiming menu or context menu.

Note

It is not possible to make smooth transition into or from freeze frame.

Add Speed Transition and Slide

Reference

Editor:: Video Sequencer

Menu::

Strip ▶ Retiming ▶ Add Speed Transition and Slide

It is possible to create smooth transition from one speed to another speed. This can be done by selecting retiming key between 2 segments of different speeds, and choosing Add Speed Transition either from strip retiming menu or context menu. This will create 2 keys, that are linked and always move in opposite direction. If both keys are moved at once, this changes where transition starts and ends.

Delete Retiming Keys

Reference

Editor:: Video Sequencer

Menu:: *Strip ▶ Retiming ▶ Delete Retiming Keys*

Retiming key can be deleted by selecting and pressing Delete or X. When handle is deleted, strip size will not change, therefore speed will change to average between 2 retimed segments.

Note

When transition key is removed, it will re-create simple retiming key from which transition was created.

Reset Retiming

Reference

Editor:: Video Sequencer

Menu:: *Strip ▶ Retiming ▶ Reset Retiming*

Reverts all timing to the original strip.

Set Speed

Reference

Editor:: Video Sequencer
Menu:: *Strip ▶ Retiming ▶ Set Speed*

Sets the speed of a retimed segment.

Speed

The rate compared to the original time.

Preserve Current retiming

Keeps the speed of the other retiming segments unchanged by adjusting the [Duration](#) of the strip instead.

Toggle Retiming Keys

Reference

Editor:: Video Sequencer
Menu:: *Strip ▶ Retiming ▶ Toggle Retiming Keys*
Shortcut:: Ctrl-R

Enable's the [Show Retiming Keys](#) strip property. This allows retiming keys to be shown for the first time and enables interacting with them.

Meta Strips

A Meta Strip is a strip which contain multiple strips treated as if it was one strip. It allows you to reduce the vertical space used in the Sequencer. You can edit it the same way as any other strips.

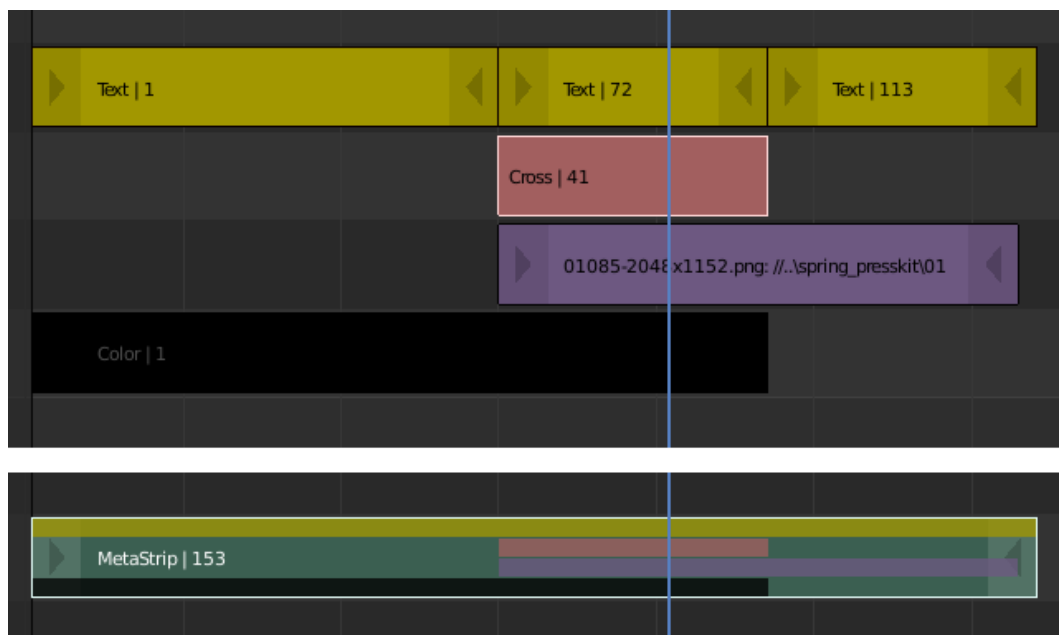
It is organization tool. For example, if you are using a lot of strips with complicated arrangement, you can group them together using Meta strips.

Make Meta Strip Ctrl-G

To create a Meta strip, select all the strips you want to group, and Ctrl-G to group them. The Meta strips will span from the beginning of the first strip to the end of the last one, and condenses all channels into a single strip.

UnMeta Strip Ctrl-Alt-G

Separating (ungrouping) the Meta strip restores the strips to their relative positions and channels. This can be used if you choose to delete a Meta strip and want to keep the strips inside.



Example of Meta strips.

You can edit the content inside a Meta strip by pressing Tab. It will expand the strip to the whole view and hide any other strips. To exit the Meta strip press Tab again. Meta strips can also be nested, which make editing them a little confusing. To exit out one level of Meta Strip make sure you do not have a Meta strips selected when you press Tab.

Note

The default blend mode for a Meta strip is Replace. There are many cases where this alters the results of the animation so be sure to check the results and adjust the blend mode if necessary.

One convenient use for Meta strips is when you want to apply the same effect to multiple strips. For example: if you have a video that was recorded in different files and want to add an effect strip. It is much more convenient to apply a single set of effects to one Meta strip than applying it to each individual strip.

See also

It is also possible to do the similar task described above with an [Adjustment Layer](#) effect strip.

Assets, Files, & Data System

- [Introduction](#)
 - [Outliner](#)
- [Blender File](#)
 - [Opening & Saving](#)
 - [Compatibility](#)
 - [Packed Data](#)
 - [Blend-Files Previews](#)
 - [Rename](#)
- [Data-Blocks](#)
 - [Data-Block Types](#)
 - [Life Time](#)
 - [Name & Rename](#)
 - [Sharing](#)
 - [Making Single User](#)
 - [Removing Data-Blocks](#)
- [Custom Properties](#)
 - [Editing Properties](#)
- [Linked Libraries](#)
 - [Link & Append](#)
 - [Library Overrides](#)
- [Asset Libraries](#)
 - [Introduction](#)
 - [Asset Catalogs](#)
- [Media Formats](#)
 - [Supported Graphics Formats](#)
 - [Supported Video & Audio Formats](#)
- [Importing & Exporting Files](#)
 - [Alembic](#)
 - [Collada \(Legacy\)](#)
 - [Universal Scene Description](#)
 - [Wavefront OBJ](#)
 - [Stanford PLY](#)
 - [STL](#)

- [Import/Export SVG as Grease Pencil](#)
- [Export Grease Pencil as \(PDF\)](#).

Introduction

Each blend-file contains a database. This database contains all scenes, objects, meshes, textures, etc. that are in the file.

A file can contain multiple [Scenes](#) and each scene can contain multiple [Objects](#). Objects can contain multiple materials which can contain many textures. It is also possible to create links between different objects, or share data between objects. A file can link data from other Blender files.

Outliner

You can easily inspect the contents of your file by using the *Outliner* editor, which displays all of the data in your blend-file.

The *Outliner* allows you to do simple operations on objects, such as selecting, renaming, deleting, linking and parenting.

[Read more about the Outliner.](#)

Blender File

- [Opening & Saving](#)
 - [Opening Files](#)
 - [Open Recent](#)
 - [Recover](#)
 - [Saving Files](#)
 - [Save Incremental](#)
 - [Save As](#)
 - [Save Copy.](#)
 - [Relative Paths](#)
- [Compatibility](#)
 - [Backward Compatibility.](#)
 - [Forward Compatibility.](#)
- [Packed Data](#)
 - [Pack Data](#)
 - [Unpack Data](#)
 - [Pack Linked Libraries](#)
 - [Unpack Linked Libraries](#)
- [Blend-Files Previews](#)
 - [Blend-File Preview](#)
 - [Data-Blocks Previews](#)
- [Rename](#)
 - [Rename Active Item](#)
 - [Batch Rename](#)

Opening & Saving

Opening and saving blend-files is usually done using the [File Browser](#).

Tip

Blend-files can also be opened by dragging and dropping blend-files into the Blender window. This method also allows to [link/append](#) the file.

Note

Unsaved Changes

By default, when exiting Blender or loading a new blend-file, if you have unsaved changes, a pop-up will ask you to either confirm discarding those changes, or save them.

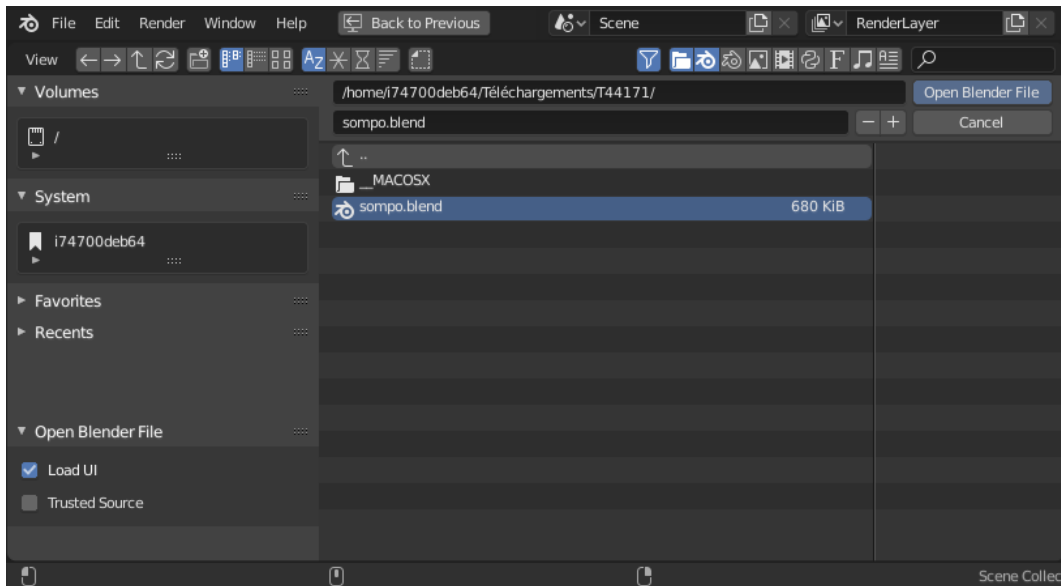
This behavior can be disabled with the *Save Prompt* option in the [Save & Load](#) section of the *Preferences*.

Opening Files

Reference

Menu::	<i>File ▶ Open...</i>
Shortcut::	Ctrl-O

The upper text field displays the current directory path, and the lower text field contains the selected filename.



The File Browser in open configuration.

Options

Load UI

When enabled, the screen layout saved inside each blend-file is used, replacing the current layout and [Workspaces](#). Otherwise the file screen layout is ignored.

Tip

If you want to work on a blend-file using your own defaults, start a fresh Blender, open the File Browser and turn off the *Load UI* button, and open the desired file.

Trusted Source

When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See [Python Security](#) to configure default trust options.

Open Recent

Reference

Menu:: *File ▶ Open Recent*

Shortcut:: Shift-Ctrl-0

Displays a list of recently opened blend-files. Hovering over items will show a preview, and information about the blend-file. Select any of the file names in the list to open that blend-file. When RMB on a listed item, a context menu will appear; One of the available options is *Open File Location*, which will open that location in an OS file explorer or Finder window.

Clear Recent Files List

Removes items from the recent files list.

Remove

Choose which type of items to remove.

All Items:: Removes all recent files.

Items Not Found:: Removes files from the list that cannot be found i.e. that have been moved or deleted.

Recover

Last Session

Reference

Menu:: *File ▶ Recover ▶ Last Session*

This will load the `quit.blend` file Blender automatically saved just before exiting. This option enables you to recover your last work [session](#) if, for

example, you closed Blender by accident.

Auto Save

Reference

Menu:: *File ▶ Recover ▶ Auto Save*

This will allow you to open an automatically saved file to recover it.

See also

[Auto Save](#)

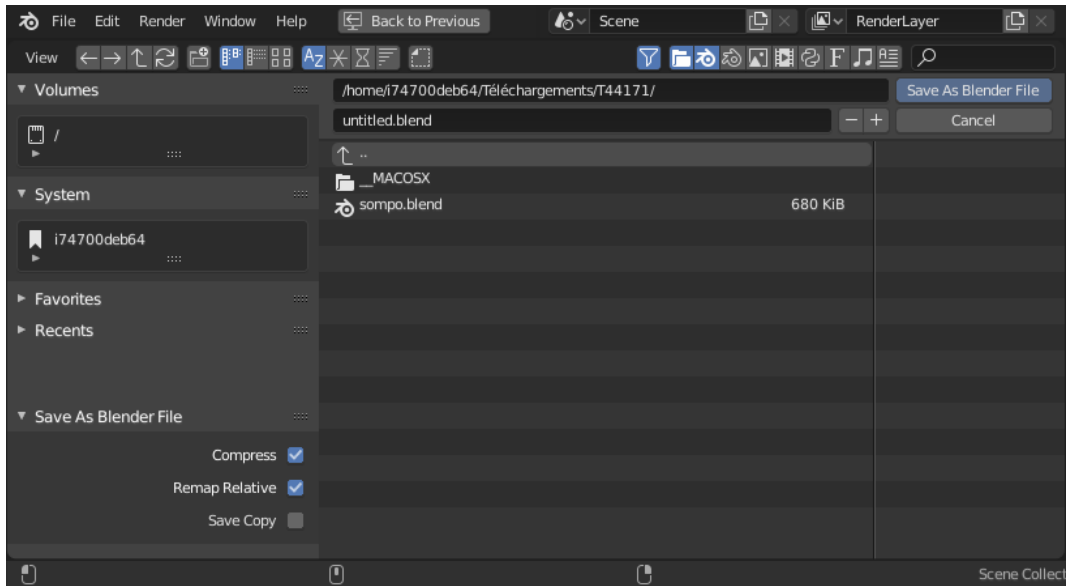
Saving Files

Reference

Menu:: *File ▶ Save*

Shortcut:: Ctrl-S

Save current blend-file over itself (if it was not saved yet, this will automatically switch to *Save As...*).



The File Browser in save configuration.

Save Incremental

Reference

Menu:: *File ▶ Save Incremental*

Shortcut:: *Ctrl-Alt-S*

Save the current Blender file with a numerically incremented name that does not overwrite any existing files.

Save As

Reference

Menu:: *File ▶ Save As...*

Shortcut:: *Shift-Ctrl-S*

Choose a file path to save the blend-file to.

Warning

If a file with the same given name already exists, the text field will turn red as a warning that the file will be overwritten.

Tip

Use the *plus* or *minus* buttons to the right of the file name, or NumpadPlus, NumpadMinus to increase/decrease a number at the end of the file name (e.g. changing file_01.blend to file_02.blend).

Options

Compress

Reduces the file size of the resulting blend-file but takes longer to save and load. This option is useful for distributing files online and saving drive space for large projects. But it can cause slowdowns when quitting Blender, or under normal operation when auto-saving backup files. See [Compression & Memory Use](#) for more information.

Hint

The used compression algorithm is Zstandard. It is not unique to Blender so files can be compressed/decompressed with external tools.

Changed in version 3.0: Prior to this version, the compression algorithm used was Gzip. This means to open newer blend-files in versions prior to 3.0, blend-files must first be saved without compression in a newer version of Blender or decompressed using an external Gzip tool.

Remap Relative

This option remaps [Relative Paths](#) (such as linked libraries and images) when saving a file in a new location.

Save Copy

Saves a copy of the actual working state but does not make the saved file active.

Save Copy

Reference

Menu:: *File ▶ Save Copy...*

Choose a file path to save the blend-file to, but return to editing the original file upon completion. This can be used to save backups of the current working state without modifying the original file.

For options see [Save As](#).

Relative Paths

Many blend-files reference external images or other linked blend-files. A path tells Blender where to look for these files. If the external files are moved, the blend-file that references them will not look right.

When you specify one of these external files, the default option is to make the path relative. Blender stores a partial path evaluated relative to the directory location of the referencing blend-file. This choice helps when you need to reorganize folders or move your files.

With a relative path, you can move the blend-file to a new location provided the externally linked files are moved along with it. For example, you could send someone a folder that contains a blend-file and a subfolder of external images that it references.

When relative paths are supported, the File Browser provides a *Relative Path* checkbox, when entering the path into a text field, use a double slash prefix (//) to make it so.

Relative paths are the default but this can be changed in the [File](#) tab of the *Preferences*.

Note

You cannot use relative paths into a new *untitled* blend-file. Save it before linking to external files.

Hint

If it is necessary to relocate a blend-file relative to its linked resources, use Blender's File [Save As](#) function which has an option to *Remap Relative* file links.

Compatibility

Blender can open blend-files saved with both older versions of the software (backward compatibility), and newer ones (forward compatibility). This comes with some limitations though.

Tip

When having issues with opening much older (or newer) blend-files, it can help to use a few intermediary Blender releases to perform conversions by smaller steps.

Note

Here is a [more exhaustive documentation](#) about compatibility handling, in the developer's documentation.

Backward Compatibility

Opening older files and converting them for the current version of Blender is usually straight-forward. It is expected to give very good and usable results.

There can be major feature changes, for which the backward compatibility will only be ensured for a limited amount of time. For example the changes to the animation system that happened during the Blender 2.5x project. This will never be less than a full major release cycle (i.e. two years at least).

Forward Compatibility

Loss of Data

Forward compatibility is inherently harder to ensure, and loss of feature should always be expected when opening a blend-file saved with a more recent version of Blender.

A warning is shown in the UI when editing a more recent blend-file. Trying to overwrite it (with a simple ‘Save’ operation) will also show a confirmation popup, as this could make that loss of data permanent.

Complete Incompatibility

When Blender switches to a new major version release (e.g. from 3.x to 4.0), there can also be major changes that will make the blend-file fully incompatible with older versions of Blender.

In such cases, older Blender will fail opening (or appending/linking from) the newer blend-file, with a message stating which minimal version is needed to open it.

In such cases, the last LTS release of the previous release cycle will be kept compatible with the newer file format version, and will be usable as converter between both versions.

For example, Blender 3.6LTS can open files from Blender 4.x, and will perform the necessary conversion such that when re-saved from 3.6, the files become compatible with all 3.x Blender versions.

Packed Data

Blender has the ability to encapsulate (incorporate) various kinds of data within the blend-file that is normally saved outside of the blend-file. For example, an image texture that is an external image file can be put “inside” the blend-file. This allows sharing a full project as a single file, instead of e.g. an archive containing the blend-file and all its dependencies.

You know that a data is packed when you see a little “gift box” icon displayed next to its path.

Warning

Not all external files can be packed

Some typically heavy external files, like videos from the [Sequence Editor](#) or [Movie Clips](#), cannot be packed in a blend-file.

Pack Data

Pack Resources

Reference

Panel:: *File ▶ External Data ▶ Pack Resources*

Mark all eligible external resource files used by the blend-file as packed. Actual packing will happen on the next save of the blend-file.

Automatically Pack Resources

Reference

Panel:: *File ▶ External Data ▶ Automatically Pack Resources*

When enabled, this option will ensure that all eligible external resource files, existing or added later, are systematically marked as packed. As with *Pack Resources*, the blend-file must be saved to the drive for this to have an effect.

Disabling that option won't unpack anything, but future external files won't be automatically marked as packed anymore.

Selective Packing

A single file can be packed by clicking on the little “gift box” icon to the left of its file-path UI widget.

Unpack Data

Unpack Resources

Reference

Panel:: *File ▶ External Data ▶ Unpack Resources*

Unpack all external resource files stored into a blend-file.

Options

Use files in current directory (create when necessary)

Unpacks all files in the same directory // as the blend-file, grouping them in proper folders (like “textures” for instance). However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to current directory (overwrite existing files)

As with previous option, but if the final file exists already, it will overwrite it.

Use files in original location (create when necessary)

Unpacks all files in their original location. However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to original location (overwrite existing files)

As with previous option, but if the final file exists already, it will overwrite it.

Disable AutoPack, keep all packed files

Only deactivates the *Automatically Pack Resources* option.

Selective Unpacking

A single file can be unpacked by clicking on the little “gift box” icon to the left of its file-path UI widget.

Options

Remove Pack

Just mark the file as unpacked, without actually writing it or reloading it from the drive.

Create <local file path>

Unpack the file at the proposed path, which is local to the current blend-file.

Use <original file path> (differs)|(identical)

If the original file path still exists, mark it as unpacked. Note that it won't be automatically reloaded from the drive. (*differs*) or (*identical*) show difference status between the packed version and the one on-drive.

Overwrite <original file path>

If the original file path still exists but differs from the packed version, mark it as unpacked and overwrite the on-drive file with the packed version.

Create <original file path>

If the original file path does not exist, mark it as unpacked and write it to drive.

Pack Linked Libraries

Reference

Panel:: *File ▶ External Data ▶ Pack Linked Libraries*

Mark all linked library files in the current blend-file as packed. Actual packing will happen on the next save of the blend-file.

Unpack Linked Libraries

Reference

Panel:: *File ▶ External Data ▶ Unpack Linked Libraries*

Unpack all used linked library files from this blend-file.

Blend-Files Previews

A blend-file can store previews, both for itself, and for some of its [data-blocks](#). You can disable writing any previews when saving a blend-file using the *Save Preview Images* setting from the [Save & Load](#) section of the Preferences.

Blend-File Preview

Blender saves by default a small preview of current scene in the blend-file. This will show in the *Thumbnail* view of the [File Browser](#).

During its installation, Blender also adds a small tool to your OS, that will allow your system file browser to show those previews as file thumbnails as well.

Data-Blocks Previews

Blender will automatically generate previews for some type of data, mainly the ones related to shading (like images, textures, materials, lights and world shaders).

It can also store previews for scenes, collections and objects, but those need to be generated manually.

These previews can then be used by the *Thumbnail* view of the File Browser, when linking or appending data-blocks.

Refresh Data-Block Previews

Reference

Menu:: *File ▶ Data Previews ▶ Refresh Data-blocks Previews*

Refresh all data-block previews that can be automatically generated by Blender (shading-related ones), in the current blend-file. You still need to save the file if you want to write them to the drive.

Batch Generate Previews

Reference

Menu:: *File ▶ Data Previews ▶ Batch Generate Previews*

Generate some data-block types' previews (you can choose which in its options), in one or more blend-files on your drive. You should not use this operator on the file currently opened in Blender.

This is currently the only way to generate and store in blend-files previews for scenes, collections and objects. Note that since this involves a lot of rendering, even of small sizes, the process may take some time to complete.

Scenes

Generate previews of scenes and their collections.

Collections

Generate previews of collections of objects.

Objects

Generate previews of objects.

Materials & Textures

Generates previews for materials, textures, images, and other internal data.

Trusted Blend Files

When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file

yourself, or you trust that the person who gave it to you did not include any malicious code with it. See [Python Security](#) to configure default trust options.

Save Backups

Keep a [backup version](#) (blend1-file) of the files when saving with generated previews.

Clear Data-Block Previews

Reference

Menu:: *File ▶ Data Previews ▶ Clear Data-blocks Previews*

Clear all, a generic type of, or a specific data-block type of previews in the current blend-file. You still need to save the file if you want to clear them from the drive.

Batch Clear Previews

Reference

Menu:: *File ▶ Data Previews ▶ Batch Clear Previews*

Clear some data-block types' previews (you can choose which in its options), in one or more blend-files on your drive. You should not use this operator on the file currently opened in Blender.

Scenes

Clear previews of scenes and their collections.

Collections

Clear previews of collections of objects.

Objects

Clear previews of objects.

Materials & Textures

Clear previews for materials, textures, images, and other internal data.

Trusted Blend Files

When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See [Python Security](#) to configure default trust options.

Save Backups

Keep a [backup version](#) (blend1-file) of the files when saving with cleared previews.

Rename

Rename Active Item

Reference

Menu:: *Edit ▶ Rename Active Item*

Shortcut:: F2

The *Rename Active Item* operator renames the active [Bone](#), [Node](#), [Object](#) and [Sequence Strip](#).

When the operator is executed, a pop-up dialog appears. The text field shows the name of the current item and can be overwritten to rename the item. Return confirms the name while Esc cancels the operator.

Batch Rename

Reference

Menu:: *Edit ▶ Batch Rename*

Shortcut:: Ctrl-F2

The *Batch Rename* operator can rename many data-block names at once. This uses a pop-up dialog with operations and their options to change the name. These actions are applied in order, from first to last.

Data Source

Where to look for the data-blocks that are intended to be renamed.

Selected:: Operates on the currently selected objects.

All:: Operates on all data in the blend file.

Data Type

The [data-block type](#) to perform the batch rename operations on.

Operations

The *Batch Rename* has several sub Operations to change the data names. The default operation is *Find/Replace* however, other operations can be added to change the data names further. Below all the operations gives a message in the status bar on how many data-blocks were renamed.

Find/Replace

Find/Replace searches for a particular text in the names and optionally replaces it with a new text. [Regular Expressions](#) can be used as a powerful way to tailor the *Find/Replace* texts and can be enabled using the icon to the right of the text fields.

Find

The text to search for in names.

Replace

The text to replace for in matching names found from the *Find* text.

Case Sensitive

Search results must exactly match the case of the *Find* text.

Set Name

Set Name works the most similar to [Rename Active Item](#) by renaming the current data-block without having to do a find and replace operation.

Method

- | | |
|-----------------|--|
| New:: | Disregards the current name replacing it with the “new” name. |
| Prefix:: | Adds text to the beginning of the current name.
This is useful for tools that look for special text in the prefix of a data-block name. |
| Suffix:: | |

Adds text to the end of the current name. This is useful for tools that look for special text in the suffix of a data-block name.

Name

Defines the new name or the text to add as a prefix/suffix.

Strip Characters

Strip Characters cleans up names by removing certain character types from either the beginning or the end of the name.

Characters

- | | |
|----------------------|--|
| Spaces:: | Strips any space characters from the name, e.g. “Living Room “ becomes “Living Room”. |
| Digits:: | Strips any numerical characters from the name, e.g. cube.001 becomes cube.. |
| Punctuation:: | Strips any punctuation characters (, . ? ! : ; etc.) from the name, e.g. cube? becomes cube. |

Tip

Multiple character types can be removed at once by Shift-LMB on the types.

Strip From

- | | |
|----------------|---|
| Start:: | Strips any leading characters in the name. |
| End:: | Strips any trailing characters in the name. |

Change Case

Change Case modifies the case of names to be one of the following:

Convert To

Upper Case

Changes all text to be in upper case, e.g. cube.001 becomes CUBE.001.

Lower Case

Changes all text to be in lower case, e.g. CUBE.001 becomes cube.001.

Title Caps

Changes all text to be in title case, e.g. living room becomes Living Room.

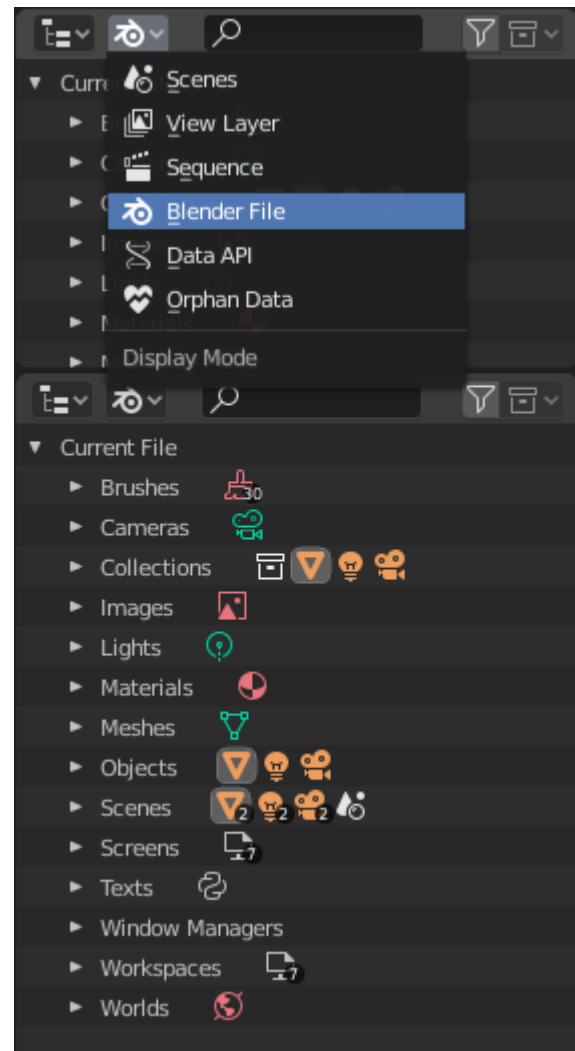
Data-Blocks

The base unit for any Blender project is the data-block. Examples of data-blocks include: meshes, objects, materials, textures, node trees, scenes, texts, brushes, and even Workspaces.

A data-block is a generic abstraction of very different kinds of data, which features a common set of basic features, properties and behaviors.

Some common characteristics:

- They are the primary contents of the blend-file.
- They can reference each other, for reuse and instancing. (Child/parent, object/object-data, materials/images, in modifiers or constraints too...)
- Their names are unique within a blend-file, for a given type.
- They can be added/removed/edited/duplicated.
- They can be linked between files (only enabled for a limited set of data-blocks).
- They can have their own animation data.
- They can have [Custom Properties](#).



Blender File view of the Outliner.

User will typically interact with the higher level data types (objects, meshes, etc.). When doing more complex

projects, managing data-blocks becomes more important, especially when inter-linking blend-files. The main editor for that is the [Outliner](#).

Not all data in Blender is a data-block, bones, sequence strips or vertex groups e.g. are not, they belong to armature, scene and mesh types respectively.

Data-Block Types

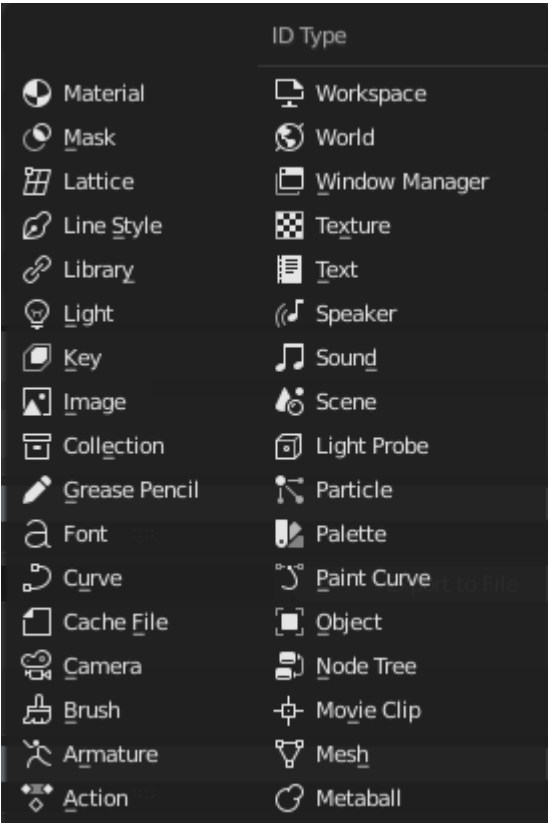
For reference, here is a table of data-blocks types stored in blend-files.

Link

Library Linking, supports being linked into other blend-files.

Pack

File Packing, supports file contents being packed into the blend-file (**not** applicable for most data-blocks which have no file reference).



Data-blocks types with their icon.

Type	Link	Pack	Description
Action	✓	—	Stores animation F-Curves. Used as data-block animation data, and the Nonlinear

Type	Link	Pack	Description
			Animation editor.
Armature	✓	—	Skeleton used to deform meshes. Used as data of armature objects, and by the Armature Modifier.
Brush	✓	—	Used as brush assets in sculpt and paint modes.
Camera	✓	—	Used as data by camera objects.
Cache File	✓	—	Used by Mesh Cache modifiers.
Curve	✓	—	Used as data by curve, font & surface objects.
Font	✓	✓	References font files. Used by curve object-data of text objects.
Grease Pencil	✓	—	2D/3D sketch data used by Grease Pencil objects. Used as overlay <i>helper</i> info, by the 3D Viewport, Image, Sequencer & Movie Clip editors.
Collection	✓	—	Group and organize objects in scenes. Used to instance objects, and in library linking.

Type	Link	Pack	Description
Image	✓	✓	Image files. Used by shader nodes and textures.
Keys (Shape Keys)	✗	—	Geometry shape storage, which can be animated. Used by mesh, curve, and lattice objects.
Light	✓	—	Used as object data by light objects.
Library	✗	✓	References to an external blend-file. Access from the Outliner's <i>Blender File</i> view.
Line Style	✓	—	Used by the Freestyle renderer.
Lattice	✓	—	Grid based lattice deformation. Used as data of lattice objects, and by the Lattice Modifier.
Mask	✓	—	2D animated mask curves. Used by compositing nodes & sequencer strip.
Material	✓	—	Set shading and texturing render properties. Used by objects, meshes & curves.
Metaball	✓	—	An isosurface in 3D space. Used as data of metaball objects.

Type	Link	Pack	Description
Mesh	✓	—	Geometry made of vertices/edges/faces. Used as data of mesh objects.
Movie Clip	✓	✗	Reference to an image sequence or video file. Used in the <i>Movie Clip</i> editor.
Node Tree	✓	—	Groups of re-usable nodes. Used in the node editors.
Object	✓	—	An entity in the scene with location, scale, rotation. Used by scenes & collections.
Paint Curve	✓	—	Stores a paint or sculpt stroke. Access from the paint tools.
Palette	✓	—	Store color presets. Access from the paint tools.
Particle	✓	—	Particle settings. Used by particle systems.
Light Probe	✓	—	Help achieve complex real-time lighting in EEVEE.
Scene	✓	—	Primary store of all data displayed and animated. Used as top-level storage for objects & animation.

Type	Link	Pack	Description
Sounds	✓	✓	Reference to sound files. Used as data of speaker objects.
Speaker	✓	—	Sound sources for a 3D scene. Used as data of speaker object.
Text	✓	✗	Text data. Used by Python scripts and OSL shaders.
Texture	✓	—	2D/3D textures. Used by brushes and modifiers.
Window Manager	✗	—	The overarching manager for all of Blender's user interface. Includes Workspaces, notification system, operators, and keymaps.
World	✓	—	Define global render environment settings.
Workspace	✗	—	UI layout. Used by each window, which has its own workspace.

Life Time

Every data-block has its usage counted (reference count), when there is more than one, you can see the number of current users of a data-block to the right of its name in the interface. Blender follows the general rule that unused data is eventually removed.

Since it is common to add and remove a lot of data while working, this has the advantage of not having to manually manage every single data-block. This works by skipping zero user data-blocks when writing blend-files.

Protected

Since zero user data-blocks are not saved, there are times when you want to force the data to be kept irrespective of its users.

If you are building a blend-file to serve as a library of assets that you intend to link to and from other files, you will need to make sure that they do not accidentally get deleted from the library file.

To protect a data-block, use the button with the shield icon next to its name. The data-block will then never be silently deleted by Blender, but you can still manually remove it if needed.

Name & Rename

Data-blocks names are unique within their namespace. A data-block namespace is defined by its type, and the blendfile it is stored in.

This means that there can be for example an Object and a Mesh named the same, but there cannot be two local objects named the same in a blendfile. However, it is possible to have one local and several linked Objects sharing the same name.

Data-block names have a fixed length of 63 bytes, i.e. 63 basic ASCII characters, or less when using diacritics or non-latin glyphs (the UTF8 encoding will then typically use more than a byte per character).

When Blender has to name a new data-block, or rename an existing one, it will check for name collisions. If a data-block with the same name already exists, the (re)named data-block will get a numeric extension added as a post-fix to its 'root name', like e.g. *.001*. The first available index is used (up to the 999 value, after that the postfix index values are simply incremented until no collision happen anymore).

In case adding the numeric suffix would make the data-block name too long, the root name part will be shortened as needed.

Blender will never rename another data-block when doing automatic naming. So e.g. when adding a new *Cube* object and there are already *Cube* and *Cube.001* local objects, the new one will be named *Cube.002*.

Local data-blocks can be renamed by the user in several places in the UI (like the ID selection widget, or the Outliner view). When renamed from the UI, the behavior in case of name collision is as follow:

- If the original root name is different than in the new requested name, the renamed data-block gets the first available numerical suffix.
 - E.g. assuming that there are three objects named *Sphere*, *Cube* and *Cube.001*, renaming *Sphere* to *Cube* will rename the data-block to *Cube.002*.
- If the original root name is the same as in the new requested name, the renamed data-block gets the requested name, and the conflicting of data-block is renamed accordingly.
 - E.g. assuming that there are three objects named *Sphere*, *Cube* and *Cube.001*, renaming *Cube.001* to *Cube* will rename the data-block to *Cube*, and the other data-block to *Cube.001*.

Sharing

Data-blocks can be shared among other data-blocks.

Examples where sharing data is common:

- Sharing textures among materials.
- Sharing meshes between objects (instances).
- Sharing animated actions between objects, for example to make all the lights dim together.

You can also share data-blocks between files, see [linked libraries](#).

Making Single User

When a data-block is shared between several users, you can make a copy of it for a given user. To do so, click on the user count button to the right of its name. This will duplicate that data-block and assign the newly created copy to that usage only.

Note

Objects have a set of more advanced actions to become single-user, see [their documentation](#).

Removing Data-Blocks

As covered in [Life Time](#), data-blocks are typically removed when they are no longer used. They can also be manually *unlinked* or *deleted*.

Unlinking a data-block means that its user won't use it anymore. This can be achieved by clicking on the “X” icon next to a data-block's name. If you unlink a data-block from all of its users, it will eventually be deleted by Blender as described above (unless it is a protected one).

Deleting a data-block directly erases it from the blend-file, automatically unlinking it from all of its users. This can be achieved by Shift-LMB on the “X” icon next to its name.

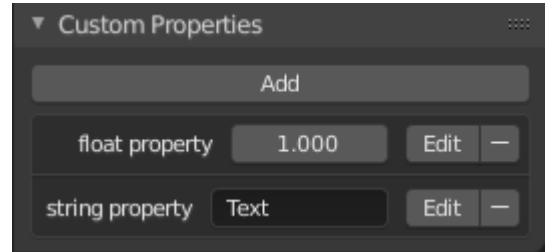
Warning

Deleting some data-blocks can lead to deletion of some of its users, which would become invalid without them. The main example is that object-data deletion (like mesh, curve, camera...) will also delete all objects using it.

Those two operations are also available in the context menu when RMB-clicking on a data-block in the *Outliner*.

Custom Properties

Custom properties are a way to store your own data in Blender's data-blocks. It can be used for rigging (where bones and objects can have custom properties driving other properties), and Python scripts, where it's common to define new settings not available in Blender. It is also possible to access custom properties from materials via the [Attribute Node](#).



Custom Properties panel.

Only certain data supports custom properties:

- All [data-blocks types](#).
- Bones and pose bones.
- Sequence strips.

To add a custom property, search for the *Custom Properties* panel, found at the bottom of most [Properties](#) or Sidebar region, and click *New*. Properties can be removed from the same location with the delete icon. Once properties are added they can be configured via the edit icon to work for a particular use case; see [Editing Properties](#) for more information.

Editing Properties

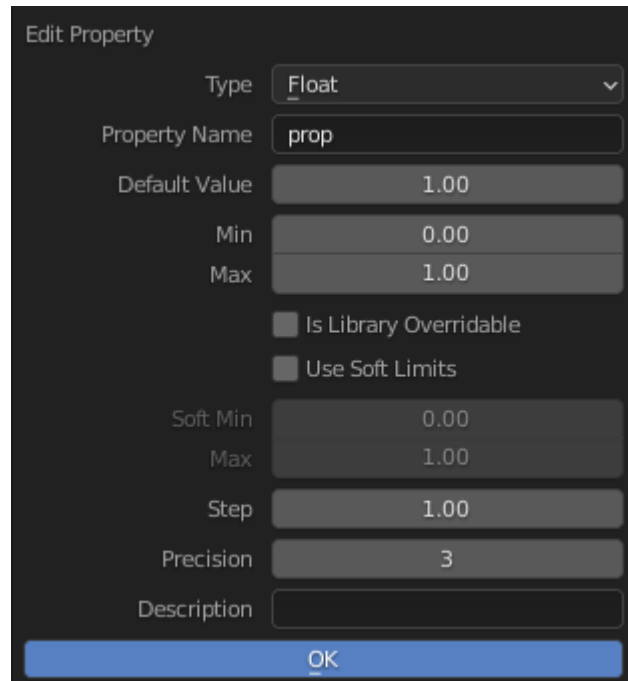
User Interface

Custom properties can be edited using the panel available for data types that support it. Editing the properties allows you to configure things such as default values, ranges, and even add a custom tooltip.

Type

The data type of the property; different data types have can only have specific data properties.

Float::



The image shows a 'Custom Properties' edit pop-up window titled 'Edit Property'. It contains several input fields for configuring a property of type 'Float'. The 'Type' is set to 'Float' in a dropdown menu. The 'Property Name' is 'prop'. The 'Default Value' is '1.00'. There are 'Min' and 'Max' fields both set to '0.00' and '1.00' respectively. There are checkboxes for 'Is Library Overridable' and 'Use Soft Limits', both of which are unchecked. Below these are 'Soft Min' and 'Max' fields, both set to '0.00' and '1.00'. There is a 'Step' field set to '1.00' and a 'Precision' field set to '3'. A 'Description' field is empty. At the bottom is a blue 'OK' button.

Custom Properties edit pop-up.

Float Array::

A numeric value with decimals e.g. 3.141, 5.0, or 6.125.

A collection of multiple float data types e.g. [3.141, 5.0, 6.125] . This data type can also be used for data that can be represented as a float array such as colors. These special float arrays can be set in the *Subtype* selector.

Integer::

A numeric value without any decimals e.g. 1, 2, 3, or 4.

Integer Array::

A collection of multiple integer data types e.g. [1, 2, 3, 4] .

Boolean::

A data type that has two possible values e.g. True or False.

Boolean Array::

A collection of boolean values e.g. [True, False, True]

String::

A sequence of characters such as “Some Text”.

Data-Block::

A reference to a Blender object, see [Data-Blocks](#).

Python::

Edit a Python data type directly, used for unsupported data types.

Array Length

The number of elements in the array. Note that if the array length is greater than 7 you cannot directly edit its elements, you must press *Edit Value* to edit the elements of the array.

Property Name

The text that is displayed to the left of the value. This name is also used to access the property via Python.

Default Value

This sets the default value of the property used by the [Reset to Default Value](#) operator.

Warning

Default values are used as the basis of [NLA blending](#), and a nonsensical default (e.g. 0 for a property used for scaling) on a property intended for being keyframed is likely to cause issues.

Min, Max

The minimum/maximum value the custom property can take.

Library Overridable

Allow the property to be [overridden](#) when the data-block is linked.

Soft Limits

Enables limits that the *Property Value* slider can be adjusted to without having to input the value numerically.

Soft Min, Max

The minimum/maximum value for the soft limit.

Step

A multiplier to control how much the data type is incremented at a time. The internal step size for floats is 0.01, so a *Step* value of 5 will

increment at a rate of 0.05 and a *Step* value of 100 will increment by 1.0. For integers the internal step size is 1.

Precision

The number of digits after the decimal to display in the user interface for float data types.

Subtype

Specifies the type of data the property contains, which affects how it appears in the user interface. This option is only available for float properties and has different options for regular floats and float arrays. Note, the unit often depends on the [Scene Units](#).

For regular floats:

Plain Data::	Data values do not have any special behavior.
Pixel::	A measure digital image resolution.
Percentage::	The displayed value is a percentage, typically you will want the Min and Max values to be 0 and 100.
Factor::	A percentage between an upper and lower bound which typical have a numerical significance.
Angle::	A measure between intersecting lines.
Time::	Time specified in seconds.
Distance::	Measure of space between items.
Power::	Work as a factor of time, measured in watts. This is used in Blender to measure light intensity.
Temperature::	Intensity of heat present.
Wavelength::	The distance between cycles of a wave measured in millimeters (mm), micrometers (μm), nanometers (nm), or picometers (pm).

For float arrays:

Plain Data::	Data values do not have any special behavior.
Linear Color::	Color in linear color space.
Gamma-Corrected Color::	Color in gamma corrected color space.

Euler Angles:: [Euler Rotation](#) angles.

Quaternion Angles:: [Quaternion Rotation](#) angles.

Note

For either of the color subtypes to work as expected the *Property Value* must be a vector with three or four values depending on the availability of an [Alpha Channel](#).

ID Type *Data-Block*

The ID-block type. For example: Key, Image, Object, Material. See [Data-Block Types](#) for a full list.

Description

Allows you to write a custom [Tooltip](#) for your property.

Python Access

Custom properties can be accessed in a similar way to [dictionaries](#), with the constraints that keys can only be strings, and values can only be strings, numbers, arrays of such, or nested properties.

See the [API documentation](#) for details.

Linked Libraries

- [Link & Append](#)
 - [Link](#)
 - [Append](#)
 - [Library Reload & Relocate](#)
 - [Make Local](#)
 - [Known Limitations](#)
- [Library Overrides](#)
 - [Override Hierarchies](#)
 - [Animation & Overrides](#)
 - [Resyncing Overrides](#)
 - [Non-Editable Overrides](#)
 - [Make an Override](#)
 - [Reset an Override](#)
 - [Clear an Override](#)
 - [Edit an Override](#)
 - [Troubleshoot an Override Hierarchy](#)

Link & Append

These functions help you reuse materials, objects and other [data-blocks](#) loaded from another blend-file. You can build libraries of common content and share them across multiple referencing files.

Newly added collections types are available in *Add ▶ Collection Instance* in the 3D Viewport.

Look in the *Outliner*, with display mode set to *Blender File*, to see all your linked and appended data-blocks.

Note

It is not possible to Append or Link data from [much newer blend-files](#)

Link

Reference

Editor::	Topbar
Mode::	All Modes
Menu::	<i>File ▶ Link</i>

Link creates a reference to the data in the source file such that changes made there will be reflected in the referencing file the next time it is reloaded. But linked data is usually not editable. [Library Overrides](#) can be created from linked data to allow some level of local editing, animation, etc.

In the [File Browser](#), navigate to the external source blend-file and select the data-block you want to reuse.

When you link an object, it will be placed in your scene at the 3D cursor position. Many other data types, cameras, curves, and materials for example, must be linked to an object before they become visible.

Options

Relative Path

See [Relative Paths](#).

Select

Makes the object *Active* after it is loaded.

Active Collection

The object will be added to the active collection of the active view layer. Otherwise, it will be added to a new collection in the active view layer.

Instance Collections

This option instantiates the linked collection as an object, adding it to the active scene. Otherwise, the linked collection is directly added to the active view layer.

Instance Object Data

Create instances for object data which are not referenced by any objects.

Append

Reference

Editor::	Topbar
Mode::	All Modes
Menu::	<i>File ▶ Link</i>

Append makes a full copy of the data into your blend-file, without keeping any reference to the original one. You can make further edits to your local copy of the data, but changes in the external source file will not be reflected in the referencing file.

In the [File Browser](#), navigate to the external source blend-file and select the data-block you want to reuse.

Tip

Blend-files can also be linked/appended by dragging and dropping blend-files into the Blender window.

Note

Appending data you already have linked will add objects/collections to the scene, but will keep them linked (and un-editable).

This is done so existing relationships with linked data remain intact.

Options

Select

Makes the object *Active* after it is loaded.

Active Collection

The object will be added to the active collection of the active view layer. Otherwise, it will be added to a new collection in the active view layer.

Instance Collections

This option instantiates the linked collection as an object, adding it to the active scene. Otherwise, the linked collection is directly added to the active view layer.

Instance Object Data

Create instances for object data which are not referenced by any objects.

Fake User

Defines the appended data-block as [Protected](#).

Localize All

Appends also all indirectly linked data, instead of linking them.

Library Reload & Relocate

Reloading is useful if you changed something in the library blend-file and want to see those changes in your current blend-file without having to re-open it. You can reload and relocate a whole library from the context menu of the library items in the *Outliner's Blender File* view,

Relocating allows you to reload the library from a new file path. This can be used to either fix a broken linked library (e.g. because the library file was moved or renamed after linking from it), or to switch between different variations of a same set of data, in different library files.

Broken Library

While loading a blend-file, if Blender cannot find a library, it will create placeholder data-blocks to replace missing linked ones. That way, references to the missing data is not lost, and by relocating the missing library, the lost data can be automatically restored.

Make Local

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	<i>Object ▶ Relations ▶ Make Local...</i>

Reference

Editor::	Outliner
Menu::	<i>Context menu ▶ ID Data ▶ Make Local</i>

Makes the selected or all external objects local to the current blend-file. Links to the original library file will be lost, but it will make those data-blocks fully editable, just like the ones directly created in that blend-file.

Options

The operation available from the *Outliner*'s context menu has no options, and only affects the selected data-block.

The operation available from the *3D Viewport* only directly affects selected objects, but it can also make local the objects' dependencies:

Type

Optionally unlinks the object's Object Data and Material Data.

Selected Objects, + Object Data, + Materials, All (i.e. including all scenes)

Known Limitations

For the most part linking data will work as expected, however, there are some corner cases which are not supported.

Circular Dependencies

In general, dependencies should not go in both directions. Attempting to link or append data which links back to the current file will likely result in missing links.

Object Rigid Body Constraints

When linking objects *directly* into a blend-file, the *Rigid Body* settings **will not** be linked in since they are associated with their scene's world. As an alternative, you can link in the entire scene and set it as a [Background Set](#).

Compression & Memory Use

Linking to blend-files with compression enabled may significantly increase memory usage while loading files.

Reading data on demand isn't supported with compression (*this only impacts load time, once loaded there is no difference in memory use*).

Library Overrides

Library Overrides is a system designed to allow editing [linked data](#), while keeping it in sync with the original library data. Most types of linked data-blocks can be overridden, and the properties of these overrides can then be edited. When the library data changes, unmodified properties of the overridden one will be updated accordingly.

Note

The old proxy system has been deprecated in Blender 3.0, and fully removed in Blender 3.2. Automatic conversion from proxies to library overrides happens when loading a blend-file, but results on complex characters are not guaranteed and may need manual fixes.

Library overrides supports:

- Multiple independent overrides of a same linked data (e.g. having the same character multiple times in the same scene).
- Adding new modifiers and constraints, anywhere in the stack.
- Recursively chaining overrides (i.e. link and override overrides from another library file, etc.).

Note

There are known issues that have to be addressed. See the [main task of the project](#), for more details.

Warning

While in most cases library overrides data is preserved across a loss of reference linked data (if e.g. the library file becomes unavailable or is relocated), there are some exceptions.

The main one is probably posed (but not animated) armature objects, when their Armature obdata itself is not overridden. The Pose bones of an armature object are fully linked to the bones of its Armature obdata, if the later goes missing, the pose bones are definitively lost.

Note

Proper Collections Layout Matters

For library overrides to work well, it is much better if **all** the collections needed by the character are children of the root (linked and instantiated) one, such that there is a clear hierarchy. Otherwise, some data may not be properly automatically overridden, and other operations may be less reliable.

Override Hierarchies

Hierarchy is a very important concept to understand when working with library overrides. In Blender, a real-life asset (a character, a prop, a set, etc.) is almost never made of a single data-block, but is rather a group of data-blocks with dependency relationships to each-other. E.g. a character will typically have an armature object, several geometry objects, rig-controllers objects, the object data for all of these objects, materials, textures, etc.

These relationships can be represented as a tree, with a root data-block 'linking-in' all its dependencies, recursively. With library overrides, typically, the root of the hierarchy is also the data-block that is directly linked when importing the asset (usually a collection).

This concept of hierarchy can also be seen as some sort of super meta-data-block. It is critical when there are several overrides of the same linked data, since it allows to clearly identify a given data-block to one override, leaving no ambiguity to processes that affect the whole hierarchy (e.g. resyncing overrides with their linked data). It also allows to share relationships

between data-blocks of different hierarchies, like a parenting relationships between two different overrides of a same character.

Animation & Overrides

Due to current design of animation data in Blender, what is editable in overrides' animations can change greatly depending on whether animation data was already defined in the linked reference data-block. Animation data is created for a datablock if it gets animated by keyframes, or through drivers.

In general, an overrides can do much more with its animation data if no animation data exists in its linked reference data-block.

Keyframes (a.k.a. F-Curves)

[Keyframed animation](#) belongs to another data-block (an Action one). So it is possible to assign a purely local Action data-block replacing the one linked from the library. This will completely replace the keyframed animation from the linked data though, and not override it in any way.

Overridden Action data-blocks only support a very limited amount of editing. For example, an existing F-Curve can be muted, but its keyframes cannot be edited, and no new F-Curve can be added.

Drivers

If the linked reference data has animation data, then its overrides only have limited possibilities to edit the existing [drivers](#). For example, it will be possible to change the existing target of a driver, but it won't be possible to add new drivers, or new targets to an existing driver.

If the linked reference data has no animation data, then its overrides will create a new one when they get some drivers defined. Drivers can then be fully edited, added or removed, just as with purely local data-blocks.

NLA

The [NLA editor](#) data also belongs to the animation data of a data-block. However, this data does support some greater level of edition in overrides, including moving or resizing existing strips from the linked data, and adding new local strips.

Resyncing Overrides

The relationships between linked data-blocks can change, resulting in outdated overrides. When this happens, overrides need to be resynced to match the new structure of their hierarchy. Overrides are automatically resynced if needed on blend-files opening. However, it may be needed to resynced them manually sometimes, see [Troubleshoot an Override Hierarchy](#).

Tip

Blender is also able to resync library overrides from external libraries, that are then linked into a working file. However, this is a costly process that needs to be fully redone every time the working file is loaded, since Blender cannot edit/modify the external library directly.

So users linking overrides (or creating recursive overrides) should ensure that their library files are regularly updated, to avoid this overhead on file load (typically, opening and saving those library files should be enough to update them).

Tip

Auto resyncing can be disabled in the [Experimental Preferences](#).

Non-Editable Overrides

For technical reasons (how relationships between data-blocks are stored), Blender needs to create overrides of a lot of data-blocks, even when only

one or two of them actually needs to be edited by the user. To reduce the amount of information and risk of potential unwanted editing, most of these data-blocks are now marked as non-editable by default. This can be changed once the override has been created.

Make an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	<i>3D Viewport ▶ Header ▶ Object ▶ Library Override ▶ Make Outliner ▶ Context Menu ▶ Library Override ▶ Make ID Widget ▶ Context Menu ▶ Library Override ▶ Make</i>
Shortcut::	Shift-LMB on the ‘linked’/‘overridden’ button of an ID Widget.

Create overrides from the selected data-blocks.

Blender automatically create overrides for all required data-blocks to ensure that valid override hierarchies are created.

Only overrides created from selected items will be user-editable.

Warning

The support for the creation of library overrides from the ID Widget (mainly from within the *Properties* editor) is limited. While the most common usages should be supported, especially with Objects, meshes, etc., much remains to be implemented.

Selected Items

Depending on where from the override is created, there are several ways to ‘select’ items to be overridden and user-editable.

Note

This also applies to the other common operations (*Reset* and *Clear*).

The *Troubleshoot* advanced operations only available from the Outliner always apply to a whole override hierarchy.

3DView

The selected objects will be considered as selected.

When a selected object is a local *Empty* instantiating a linked collection, the following will happen: * The *Empty* object will be removed. * Its linked collection will be overridden, and that override will be instanced in the same collection in the current *View Layer*. * If the collection contains *Armature* objects, they will be user-editable. Otherwise, no created override will be defined as user-editable.

Outliner

The operation can be applied on either the selected items only, their content only, or both.

Tip

Using *Selected & Content* is an easy way to get all newly created overrides immediately user-editable.

ID Widget

Only the linked data-block in the ID Widget is considered as selected, and set as editable once overridden.

Make Editable

That same operation can also be used to make existing overrides user-editable, after they have been created, or [cleared](#)

Reset an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	<i>3D Viewport ▶ Header ▶ Object ▶ Library Override ▶ Reset Outliner ▶ Context Menu ▶ Library Override ▶ Reset ID Widget ▶ Context Menu ▶ Library Override ▶ Reset</i>

Reset the selected overrides to their original values (from the linked reference data). Unlike with the *Clear* operation, the overrides remain fully editable, and are never deleted.

Clear an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	<i>3D Viewport ▶ Header ▶ Object ▶ Library Override ▶ Clear Outliner ▶ Context Menu ▶ Library Override ▶ Clear ID Widget ▶ Context Menu ▶ Library Override ▶ Clear</i>
Shortcut::	Shift-LMB on the ‘overridden’ button of an ID Widget.

Reset the selected overrides to their original values, and if possible without breaking the existing hierarchy, delete them and replace them by their linked reference data. Otherwise, keep the overrides but mark them as non-editable.

Edit an Override

Essentially, an override is edited the same way as a regular local data-block. You can use operators on them, edit their properties from various editors, etc. There are some limitations however, most notably Edit Mode is not allowed for overrides. In most cases, as soon as you edit a property, you can see that it's overridden by its teal blue outline/background.

You can also animate overrides, animated properties just replace/supersede overrides then. Note that you cannot override/edit an existing animation, you'll have to create a new action. You can manually define or remove an override from the context menu of the relevant property. If an override is not editable, you have to make it editable first.

Define Overrides

Reference

Editor::	Any
Mode::	Object Mode
Property::	<i>Context Menu ▶ Define Overrides Context Menu ▶ Define Override</i>

Mark a property to be overridden in the local blend file. For array properties all elements will be overridden.

Define Single Override

Reference

Editor::	Any
-----------------	-----

Mode:: Object Mode
Property:: *Context Menu ▶ Define Single Override*

Mark a property to be overridden in the local blend file. For array properties only the selected element will be overridden.

Remove Overrides

Reference

Editor:: Any
Mode:: Object Mode
Property:: *Context Menu ▶ Remove Overrides Context Menu ▶ Remove Override*

Remove the property from the overrides. The value of the linked in data-block will be used. For array properties all elements will be removed from the override.

Remove Single Override

Reference

Editor:: Any
Mode:: Object Mode
Property:: *Context Menu ▶ Remove Single Override*

Remove the property from the overrides. The value of the linked in data-block will be used. For array properties only the selected elements will be removed from the override.

Troubleshoot an Override Hierarchy

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	<i>Context Menu ▶ Library override ▶ Troubleshoot</i>

These operations are only available from the Outliner contextual menu. They can help fixing a broken override hierarchy.

Resync

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	<i>Context Menu ▶ Library override ▶ Troubleshoot ▶ Resync</i>

The hierarchy of the linked data (the relationships between linked data-blocks) can change. Overrides need to be resynced to match the new hierarchy. This operator will resync the override to match the new hierarchy in the library.

Warning

While resyncing a library override it is possible that edited overrides get deleted if they are changed in the original library. If this is the case, a warning message will be displayed stating how many overrides were deleted, if the deletion is undesirable the resync can be undone before saving the blend-file.

Note

This Process is Automatic

Usually, this operation happens automatically when blender detects it is needed, on file load, unless it is disabled in the [Experimental Preferences](#).

Resync Enforce

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	<i>Context Menu ▶ Library override ▶ Troubleshoot ▶ Resync Enforce</i>

In some cases, especially with older blend-files that were saved with ‘broken’ (non-hierarchy-matching) overrides, a regular resync itself cannot rebuild properly the override as expected (e.g. some objects might go missing). To solve this issue, this operator rebuilds the local override from its linked reference, as well as its hierarchy of dependencies, enforcing that hierarchy to match the linked data (i.e. ignoring existing overrides on data-blocks properties). This is similar to a regular resync, but is more forceful, aggressive, at the cost of a potential loss of some overrides on ID pointers properties.

Delete

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	<i>Context Menu ▶ Library override ▶ Troubleshoot ▶ Delete</i>

Remove the whole library override hierarchy, and replace all of these override data-blocks by their original linked data-blocks. This fully reverts the *Make* operation.

Asset Libraries

- [Introduction](#)
 - [What is an Asset?](#)
 - [What is an Asset Library?](#)
 - [Asset Types](#)
 - [The Current File Asset Library.](#)
 - [Life Cycle of an Asset](#)
 - [Bundled Assets](#)
 - [Asset System Files \(.asset.blend Extention\).](#)
 - [Design Limitations](#)
 - [Future Development](#)
- [Asset Catalogs](#)
 - [The Home Location of Assets](#)
 - [Creating Catalogs](#)
 - [Assigning an Asset](#)
 - [Saving Catalogs](#)
 - [Components of a Catalog](#)
 - [Catalog Definition Files](#)

Introduction

This section describes Blender’s asset library system. It was introduced in Blender 3.0, and will be improved and expanded over multiple upcoming releases.

See also

[Asset Browser](#)

The main interface for organizing and using assets.

[Asset Catalogs](#)

For organizing assets.

[Pose Library](#)

Built on top of the Asset Browser.

What is an Asset?

An asset is a data-block with meaning.

A blend-file is a database with multiple [Data-Blocks](#): objects, textures, materials, etc. When planning to re-use or share these, the data needs a meaning. What is this? What is this for? **Assets are curated data-blocks that are meant for easy reuse.**

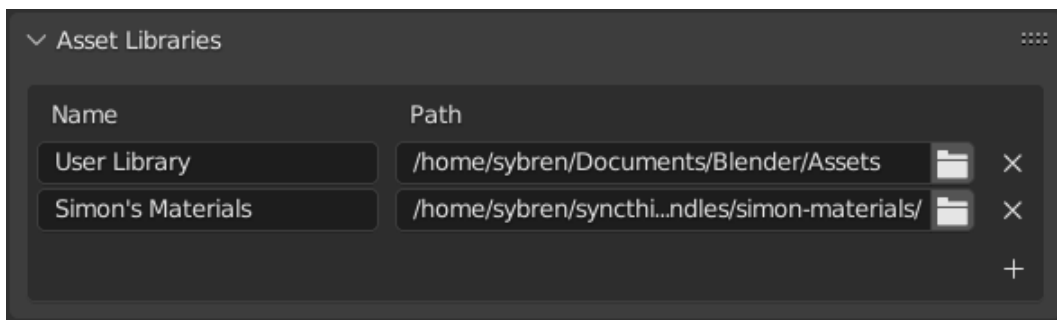
Note

The general term “asset” often also refers to other file types, such as images, sounds, video files, etc. These are currently not supported as asset in Blender.

For more info, see [Future Development](#).

What is an Asset Library?

An asset library is a directory on your drive that is registered in the Preferences as an asset library. Registering it means that you give the library a name (like “Sprite Fright”) and the location on drive (like /home/sybren/projects/sprite-fright/assets).



Name and Location of asset libraries in the Preferences.

Once registered, you can select the asset library in the Asset Browser. All the blend-files in the asset library will be scanned for assets, and all those assets will be shown in the Asset Browser.

Note

Loading an asset library for the first time may take a while, but the next time it is loaded should be significantly faster. Blender generates an index of all assets contained inside an asset library, and keeps it up-to-date as files are modified within it. The indices are stored in the [Local Cache Directory](#).

The blend-files can be directly in the top-level directory of the asset library, or in any subdirectory. The on-drive organization of asset libraries is all up to you. Regardless of which blend-file contains the assets, each asset can be

assigned a **catalog**. For more info about how to organize your assets this way, see [Asset Catalogs](#).

Asset Types

Assets can be broadly divided into two types: **primitive** and **preset** assets. Which is which depends on the [Data-Blocks](#) type.

Primitive assets are data-blocks that are either **linked or appended** to the current file. Examples are objects, materials, and worlds. These can be dragged from the Asset Browser into the scene (objects and worlds), or onto existing objects (materials).

Preset assets are data-blocks that are loaded and then **applied** to something or **activated**. An example is a pose asset. When applying the pose, the data-block is loaded from its blend-file, and then the pose is applied to the active armature. Brush assets are an example of an asset type that is activated. They get loaded into the current file and activated for painting or sculpting, but don't get saved in the file.

In the future, the asset type definition will be expanded; see [Future Development](#) for more info.

The Current File Asset Library

To help with the management of assets in the current blend-file, you can set the Asset Browser to show the **Current File asset library**. This always shows the assets in the current file, even when the current file is not saved in an asset library. This also makes it possible to create assets and use them in the same file, for small single-file projects.

When the current blend-file is part of an asset library, you can also see its assets in that library, of course. The assets that are in the current file are marked with an icon; only those are editable.

Life Cycle of an Asset

This section describes how to create, edit, share, and use assets.

Creating an Asset

To create an asset, first create the thing you want to turn into an asset. That is, create the object, material, world, or pose your character. The next step depends on the type of asset (see [Asset Types](#) above).

For primitive assets, use the **Mark as Asset** operator. It can be found in the data-block selector, in the Outliner, and for objects in the 3D Viewport Object menu. When using *Mark as Asset*, an automatic preview is generated. If you want, you can also change or replace this with an image of your own choosing; use the folder button next to the preview image in the Asset Details region of the Asset Browser.

For preset assets, there will be a dedicated button for the different asset types. For example for poses there is a **Create Pose Asset** button in the Action editor. Brush assets are created by using **Duplicate Asset** from existing brush assets.

After creating the asset, make sure the current blend-file is saved in your asset library. Blender does not copy the asset into the asset library for you.

Editing Assets

Since assets are regular data-blocks, with just a little bit of metadata attached, they can be edited like any other Blender data. Just open the file and edit the object, material, world, etc.

For poses assets, this is also possible. With the pose library file open, just click the **Assign Action** button to assign the pose action to the currently selected armature. Then you can use all of the animation tooling to edit the pose, remove or add keys, etc.

Editing asset metadata can be done via the [Asset Browser](#).

Sharing Assets

Because assets are simply stored in blend-files, they can be shared by sharing their blend-file. Be sure to include the [Asset Catalog Definition File](#) as well.

There is currently no functionality to extract selected assets and save them (together with their catalog definitions) into a different blend-file. This could be implemented as an add-on.

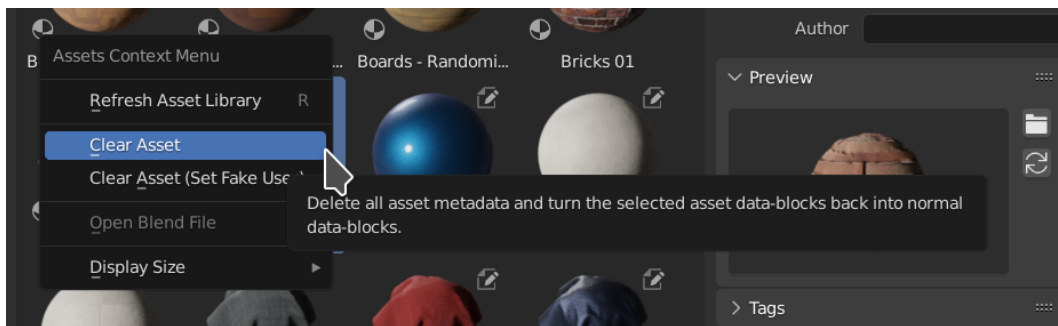
Using Assets

Assets can be used from the [Asset Browser](#).

The pose library extends this, and adds an Asset View to the 3D Viewport. See [Use from 3D Viewport](#).

Removing Assets

Asset metadata can be erased by the *Clear Asset* operator. This operator is available in data-block selectors, the Asset Browser, and for objects in the 3D Viewport menu.



Clear Asset in the Asset Browser.

Clear Asset

Removes the asset metadata (catalog, description, author, tags), effectively turning an asset into a regular data-block. As such, the same removal rules apply as with other data-blocks. For example, if a mesh object is still placed in the scene, *Clear Asset* will not remove it from

the scene. See [Life Time](#). The preview will be kept inside the data-block and not be removed.

Clear Asset (Set Fake User)

Performs the same operation as *Clear Asset*, and then marks the data-block as [protected](#). This makes it possible to no longer have the data-block marked as asset, and still be sure it is not lost when saving the blend-file.

Bundled Assets

Blender includes many assets out of the box, these are contained in the “Essentials” library.

Included in this library are:

- [Hair node groups](#)
- [Smooth By Angle Node Group](#)
- Brushes:
 - [Mesh Sculpt](#)
 - [Curve Sculpt](#)
 - [Texture Paint](#)
 - [Vertex Paint](#)
 - [Weight Paint](#)

Asset System Files (*.asset.blend* Extention)

Some types of assets can be edited without having to open a blend-file inside of an asset library. Blender saves these assets to libraries in special files using the *.asset.blend* extension. They are entirely managed by Blender’s asset system, and only contain a single asset and its dependencies.

It is still possible to save a normal file with the *.asset.blend* extension. This will then not be treated as an asset system file, Blender knows the difference.

Asset system files have one more special characteristic: You can open, but not save them. The [Save As](#) operator can still be used to create a new file from them, which will then be just a normal blend-file. Thus contained assets cannot be edited without opening the file itself. Blender shows some clear warnings to communicate that asset system files cannot be changed and saved the normal way.

The reason these files are special is that the asset system might need to regenerate them. Any additional changes done by the user might be lost then. To prevent this data-loss, these files are protected from user modifications.

Currently, only brush assets support this feature.

Design Limitations

Blender is **not allowed to write to other blend-files** than the one you have currently open, or the special *.asset.blend* files explained above. This means that to edit an asset, you have to open its blend-file. Fortunately this is only a single click away, both in the Source List region of the Asset Browser and in the asset context menu.

Future Development

This section describes interesting avenues for further development. Even though it is not an exhaustive list, it might help to better understand the current functionality of Blender's Asset Browser.

Non-Data-Block Assets

Non-Blender assets, such as image or audio files, will likely be supported in a future version. For such files, asset metadata is then stored in XMP sidecar files, similar to what other software is also doing. Importers (USD, glTF, FBX, ...) could add support for their file types as assets this way too. Furthermore, it should become possible to enrich an asset with a Python script, which can then provide code to be run when the asset is used.

Cross Blend-File Editing

As described above, Blender itself is not allowed to write to other blend-files than the currently open one. This rule helps to limit complexities; for example, it is hard to reliably implement an undo system when manipulating other files. The rule does get in the way of mass-updating assets when they are stored in various blend-files.

Since there is already tooling that can manipulate blend-files outside of Blender itself (see [Blender Asset Tracer](#)), it's possible to also create an external tool for doing such edits across blend-files. Such a tool might even be implemented via Blender's application templates system, or as an add-on; the rule above applies to Blender itself, not to its add-ons.

Asset Pushing

Note

The introduction of Brush assets in Blender 4.3 includes support for an asset pushing concept as described here. This might be brought to more asset types in future.

Asset **pushing** is a way of getting assets into the asset library, where you are working on a file and want to copy the asset from that file into the library. This is a concept that appears deceptively simple. In certain cases it is actually simple, but often enough it gets quite complex. For example, when you want to push an object into an external asset library, should that also copy the materials? What about the texture images referenced by those materials? What about objects referenced by custom properties, constraints, or modifiers? And in which files would they have to go? Do they all go into one big `assets.blend`, individual blend-files, or into a directory per asset type? Blender should not be making such decisions for you.

For specific cases, these things are all solvable. For this reason the pose library has been created as an add-on which is enabled by default. Studios with specific needs can disable the add-on and implement their own

functionality; the building blocks are all in Blender's core, and thus do not need to be copied for this. Furthermore, add-ons can write to other blend-files, so they could make the decisions for users.

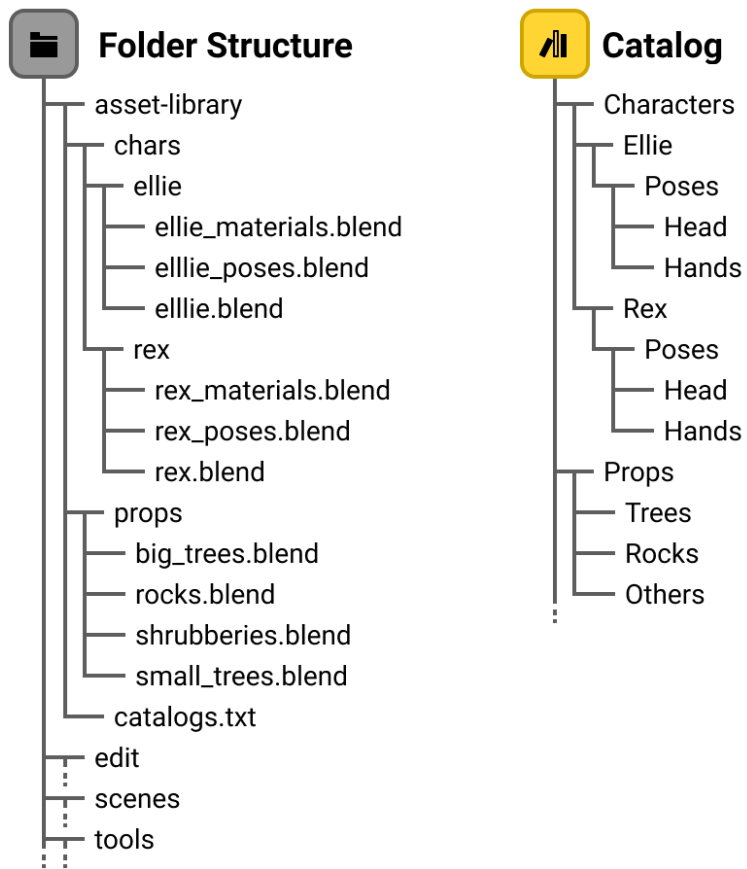
Asset pushing is desirable. Because of the questions above, it is unknown how to implement this well, in a way that still allows artists control over their assets.

Asset Catalogs

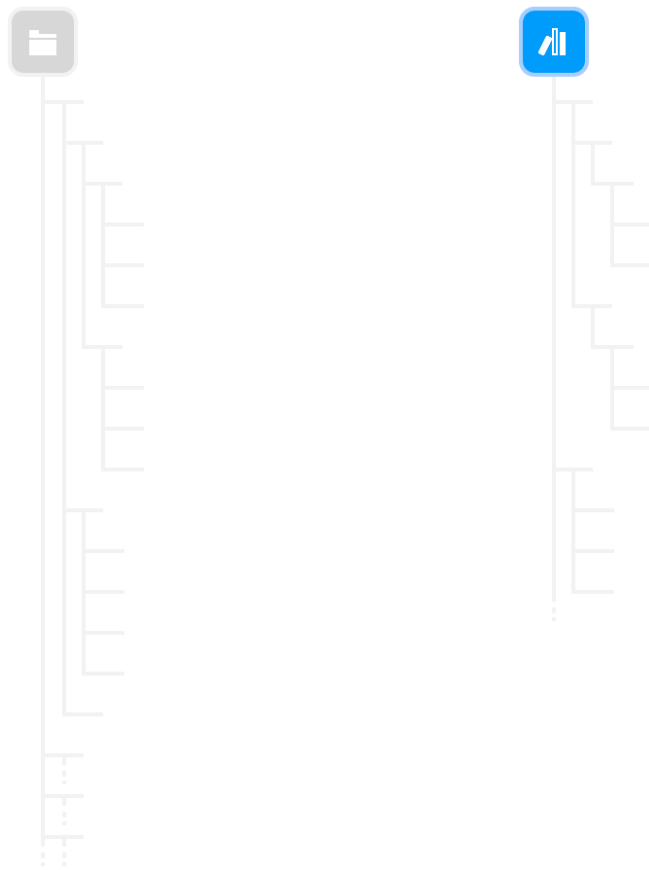
Asset Catalogs help you to organize your assets. They look a little bit like file directories, but they are completely independent of the location of your blend-files. Assign each asset in a blend-file to its own catalog, or have one big catalog with all the assets of all the blend-files combined. It's all up to you.

Similar to [Collections](#), catalogs can be nested i.e. you can have a main catalog that contains several nested catalogs. For example, this allows you to have a catalog of assets for “Furniture” with sub-catalogs of “Tables”, “Chairs”, “Lamps”, etc...

For more technical information, see [Asset Catalogs on the Blender Developer Documentation](#).



Example file system and catalog structures.



Example file system and catalog structures.

The Home Location of Assets

There can be as many catalogs as you want, but **an asset can be assigned to a single catalog** at a time. This is similar to a file system, where a file is only in one directory (ignoring advanced things like symbolic links).

Catalogs themselves can be nested and moved by dragging and dropping. Moving a catalog will not modify the assets it contains; they will simply move along to the new location of the catalog.

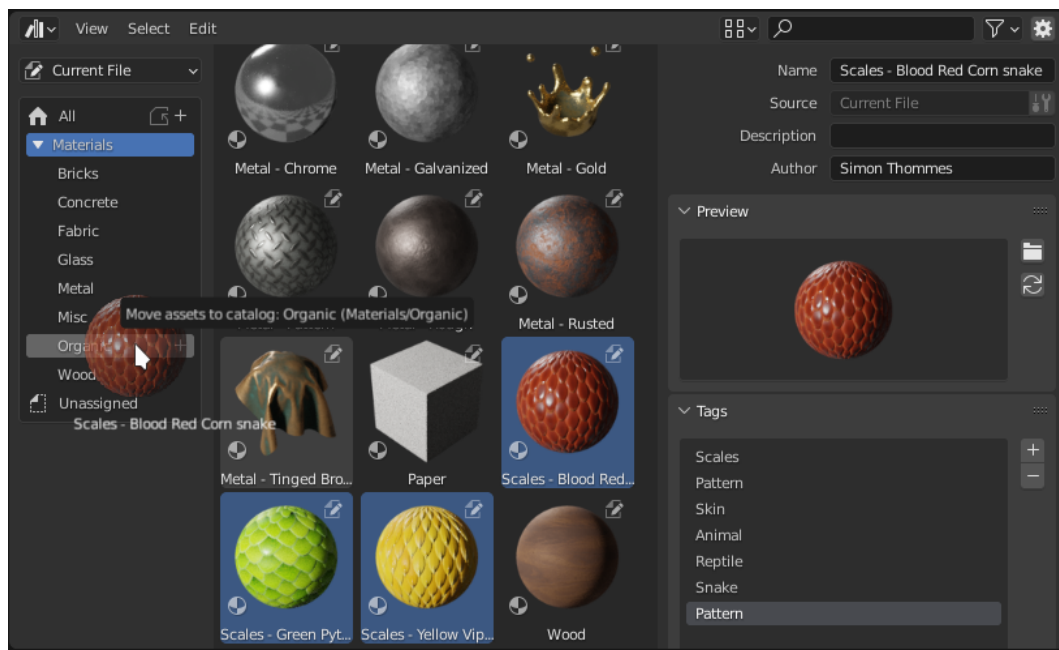
Selecting a catalog in the Asset Browser will show all assets in that catalog and in child catalogs. So, in the preceding example, selecting

Characters/Ellie/Poses will also show assets from Characters/Ellie/Poses/Head and Characters/Ellie/Poses/Hands.

Creating Catalogs

New catalogs can be created in the [Asset Browser](#) through *Header ▶ Catalog ▶ New Asset Catalog*. Once the catalog is created you can double LMB on it's name in the *Source List* region of the editor to give the catalog a more descriptive name. Catalogs can also be created in this region by clicking the plus icon found at the top of the tree view.

Assigning an Asset



Assigning a selection of “Scale material” assets to a catalog.

To assign assets to a catalog, just select and drag the assets on top of the catalog.

Tip

You can assign an asset to the “Unassigned” catalog, this will remove it from any existing catalogs.

Saving Catalogs

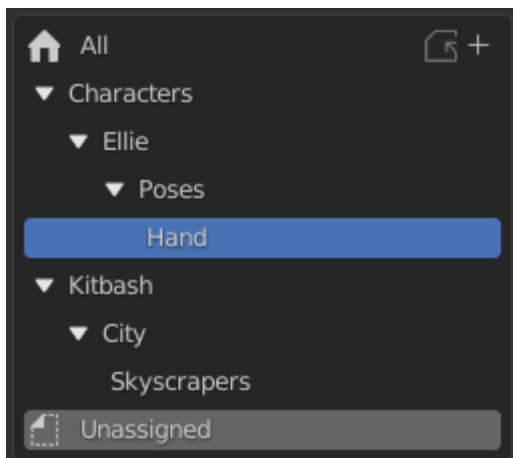
Saving catalogs makes any edits to any catalogs permanent by writing the current set up to the asset library. Catalogs can be saved in the [Asset Browser](#) through *Header ▶ Catalog ▶ Save Asset Catalog*. Once the catalog is created you can double LMB on it’s name Catalogs can also be saved in the *Source List* region of the editor by clicking the save icon found at the top of the tree view.

Components of a Catalog

Each catalog consists of a *catalog path*, a *UUID*, and a *simple name*. Normally you would only deal with the catalog path; the rest is for internal Blender use and/or for emergency situations.

Catalog Path

The path of a catalog determines where in the catalog hierarchy the catalog is shown. Examples are Characters/Ellie/Poses/Hand or Kitbash/City/Skyscrapers, which would result in the following catalog tree. The highlighted catalog has path Characters/Ellie/Poses/Hand.



Example tree of asset catalogs.

UUID

Each catalog has a [UUID](#), which is normally hidden from the user interface (enable Developer Extras and the experimental Asset Debug Info option to see them). This is what is stored in the asset, and what determines the “identity” of the catalog. As a result, a catalog can be renamed or moved around (i.e. you can change its path), and all assets that are contained in it will move along with it. This only requires a change to the catalog itself, and not to any asset blend-file.

Simple Name

Each catalog has an optional *simple name*. This name is stored along with the UUID in each asset. The purpose is to make it possible for humans to recognize the catalog the asset was assigned to, even when the *catalog definition file* (see below) is lost.

Like the UUID, the simple name is normally hidden from the user interface. Enable Developer Extras in the interface preferences to make it visible in the Asset Browser.

Catalog Definition Files

Asset catalogs are stored in Catalog Definition Files (CDFs). Blender 3.0 supports a single CDF per asset library. It is stored in `blender_assets.cats.txt` in the root directory of the asset library. If the file does not exist, Blender will create it when the catalogs are saved. When catalogs are changed, Blender updates that file, but also creates a backup of the previous state to a file named `blender_assets.cats.txt~`.

Which File to Write To

Asset catalogs can be saved independently of the blend-file; the catalog editor has its own “Save” button.

Format

Catalog Definition Files (CDFs) are relatively simple text files, encoded in UTF-8. Each CDF consists of a version indicator, and a line of text per catalog. Each catalog line is colon-separated, of the form {UUID}:{path}:{simple name}.

Example

This is an example of a valid catalog definition file:

```
# This is an Asset Catalog Definition file for Blender.  
#  
# Empty lines and lines starting with `#` will be ignored.  
# The first non-ignored line should be the version indicator.  
# Subsequent lines are of the format  
"CATALOG_UUID:catalog/path/for/assets:simple catalog name"
```

VERSION 1

```
313ea471-7c81-4de6-af81-  
fb04c3535d0e:catalog/without/simple/name:  
ee9c7b60-02f1-4058-bed6-  
539b8d2a6d34:character/Ellie/poselib:character-Ellie-poselib  
cd66bf52-58f4-45cb-a4e2-  
dc0e0ee8f3fe:character/Ellie/poselib:character-Ellie  
4eb44ec6-3424-405b-9782-  
ca006953e799:character/Ellie/poselib/white space:character-  
Ellie-poselib-white space  
b63ed357-2511-4b96-8728-  
1b5a7093824c:character/Ružena/poselib:Ružena pose library  
dcdee4df-926e-4d72-b995-  
33106983bb9a:character/Ružena/poselib/face:Ružena face  
fb698f2e-9e2b-4146-a539-  
3af292d44899:character/Ružena/poselib/hand:Ružena hands
```

Valid Catalog Paths

Catalog paths follow the following rules:

- All paths are absolute; there is no difference between /a/b and a/b.
- Only / as separator (no \; think less filesystem path and more URL).
- Not empty (it's required for a valid catalog).
- No empty components (so not a//b; a/b is fine).
- Invalid characters: :, \.
- Paths are always interpreted as UTF-8.

Media Formats

- [Supported Graphics Formats](#)
 - [Image Formats](#)
 - [Opening Images](#)
 - [Saving Images](#)
 - [Format Details](#)
- [Supported Video & Audio Formats](#)
 - [FFmpeg Containers](#)
 - [FFmpeg Video Codecs](#)
 - [FFmpeg Audio Codecs](#)
 - [Known Limitations](#)

Supported Graphics Formats

Image Formats

This is the list of image file formats supported internally by Blender:

Format	Channel Depth	Alpha	Metadata	DPI	Extensions
BMP	8bit	✗	✗	✓	.bmp
Iris	8, 16bit	✓	✗	✗	.sgi .rgb .bw
PNG	8, 16bit	✓	✓	✓	.png
JPEG	8bit	✗	✓	✓	.jpg .jpeg
JPEG 2000	8, 12, 16bit	✓	✗	✗	.jp2 .jp2 .j2c
Targa	8bit	✓	✗	✗	.tga
Cineon & DPX	8, 10, 12, 16bit	✓	✗	✗	.cin .dpx

Format	Channel Depth	Alpha	Metadata	DPI	Extensions
OpenEXR	float 16, 32bit	✓	✓	✓	.exr
Radiance HDR	float	✓	✗	✗	.hdr
TIFF	8, 16bit	✓	✗	✓	.tif .tiff
WebP	8bit	✓	✓	✓	.webp

Hint

If you are not interested in technical details, a good rule of thumb for selecting output formats for your project is:

Use OpenEXR

if you intend to do compositing or color grading on these images.

Use PNG

if you intend on-screen output or encoding into multiple video formats.

Use JPEG

for on-screen output where file size is a concern and quality loss is acceptable.

All these formats support compression which can be important when rendering out animations.

Hint

Bit depths for image formats represent the following numbers of tonal levels per channel:

8::	256 levels
10::	1024 levels
12::	4096 levels
16::	65536 levels

Opening Images

Relative Path

Sets the file path to be relative to the currently opened blend-file.

See [Relative Paths](#).

Detect Sequences

Automatically looks for image sequences in the selected images (based on the file name). Disable this when you do want to get single images that are part of a sequence.

Detect UDIMs

Automatically looks for [UDIM](#) tiles in the directory of the selected image; if matches are found they are loaded into Blender as UDIMs. This works by detecting if the filename has a .xxxx (four digit number) before the file extension.

Opening an Image Sequence

To load image sequence in any of the supported image file formats, the filename of the images must contain a digit to indicate the frame order (e.g. *-0001.jpg, *-0002.jpg, *-0003.jpg, etc, of any image format), indicating the frame.

The sequence could be opened by the selection of the images with any of the following methods by the confirmation with the *Open Image* button or Return.

Range

Navigate into the directory and LMB click and drag over a range of names to highlight multiple files. You can page down and continue Shift-LMB click-dragging to add more to the selection.

Batch

Shift-LMB click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (jpg, png, exr, etc.).

All

Press A to select/deselect all files in the directory.

Saving Images

File Format

Choose what format to save the image as.

Color Mode

Choose the color format to save the image (or video) to. Note that *RGBA* is not available for all image formats, check the list above for details.

BW, RGB, RGBA

Color Depth

Some image file formats support a varying number of bits per pixel. This affects the color quality and file size. Commonly used depths:

8-bit::	Most common for on-screen graphics and video.
10, 12, 16-bit::	Used for some formats focusing on photography and digital films (such as DPX and JPEG 2000).
16-bit Half Float::	Since full 32bit float is often more than enough precision, half float can save drive space while still providing a high dynamic range.
32-bit Float::	Highest quality color depth.

Note

Internally Blender's image system supports either:

- 8 bits per channel (4×8 bits).
- 32 bits float per channel (4×32 bits) – *using 4 times as much memory*.

Images higher than 8 bits per channel will be converted into a float on loading into Blender.

Compression

Used to reduce the size of the image file. How this is done may vary depending on the file format and settings used.

Quality

Similar to *Compression* but is used for JPEG based file formats. The quality is a percentage, 0% being the maximum amount of compression and 100% is no compression.

Save As Render

Save image with render [color management](#). For display image formats like PNG, apply view and display transform. For intermediate image formats like OpenEXR, use the default render output color space.

Copy

The Copy checkbox will define if the data-block will reference the newly created file or the reference will be unchanged, maintaining it with the original one.

Color Space

To specify the color space of the source file.

The list of color spaces depends on the active [OCIO config](#). The default supported color spaces are described in detail here: [Default OpenColorIO Configuration](#)

Note

Note, *Cineon*, *DPX*, *OpenEXR*, and *Radiance HDR* image types default to being saved in a linear color space.

Format Details

Cineon & DPX

Cineon is Kodak's standard for film scanning, 10 bits per channel and logarithmic. DPX has been derived from Cineon as the ANSI/SMPTE industry standard. DPX supports 16-bit colors/channels, linear as well as logarithmic. DPX is currently a widely adopted standard used in the film hardware/software industry.

DPX as well as Cineon only stores and converts the “visible” color range of values between 0.0 and 1.0 (as a result of rendering or composite).

OpenEXR

[ILM's OpenEXR](#) has become a software industry standard for HDR image files, especially because of its flexible and expandable structure.

An OpenEXR file can store multiple layers and passes. This means OpenEXR images can be loaded into a Compositor keeping render layers and passes intact.

Output Options

Available options for OpenEXR render output are:

Color Depth

Half saves images in a custom 16 bits per channel floating-point format. This reduces the actual “bit depth” to 10-bit, with a 5-bit power value and 1-bit sign.

Float (Half), Float (Full)

Codec

PXR24::	Lossy algorithm from Pixar, converting 32-bit floats to 24-bit floats.
ZIP::	Standard lossless compression using Zlib, operating on 16 scanlines at a time.
PIZ::	Lossless wavelet compression. Compresses images with grain well.
RLE::	Run-length encoded, lossless, works well when scanlines have same values.
ZIPS::	Standard lossless compression using Zlib, operating on a single scanline at a time.
DWAA::	JPEG-like lossy algorithm from DreamWorks; compresses blocks 32 scanlines together.
DWAB::	Same as DWAA but compresses blocks of 256 scanlines.

Preview

On rendering animations (or single frames via command line), Blender saves the same image also as a JPEG, for quick preview or download.

Radiance HDR

Radiance is a suite of tools for lighting simulation. Since Radiance had the first (and for a long time the only) HDR image format, this format is supported by many other software packages.

Radiance .hdr files store colors still in 8 bits per component, but with an additional (shared) 8-bit exponent value, making it 32 bits per pixel.

Supported Video & Audio Formats

Blender used [FFmpeg](#) to handle video encoding/decoding various video formats. These formats are primarily used for compressing rendered sequences into a playable movie. Video formats are composed of a container, a codec, and sometimes audio which is stored using its own codec. The roll of the container to encapsulate video and audio data that is compressed using a codec.

Codecs compress the channels of a video down to save space and enable continuous playback. *Lossy* codecs make smaller files at the expense of image quality, while *lossless* codecs compress as much as possible the video/audio, but without losing any existing data.

Some codecs, like H.264, are great for larger images. Codecs are used to encode and decode the movie, and so must be present on both the encoding machine (Blender) and the target machine. The results of the encoding are stored in a container file.

There are dozens, if not hundreds, of codecs, including Xvid, H.264, DivX, Microsoft, and so on. Each has advantages and disadvantages, and compatibility with different players on different operating systems.

Note

Most codecs can only compress the RGB or YUV colors, but some support the Alpha channel as well. Codecs that support RGBA include:

- [FFmpeg video codec #1](#)
- [PNG](#)
- QuickTime Animation
- WebM/VP9 (although Blender will not import the alpha channel due to a [limitation of FFmpeg](#)).

FFmpeg Containers

MPEG-1::

A standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to 1.5 Mbit/s. This container enforces the video codec, you can only define quality parameters, and the audio codec.
File Extensions: .mpg, .mpeg

MPEG-2::

A standard for “the generic coding of moving pictures and associated audio information”. It describes a combination of lossy video compression and lossy audio data compression methods which permit storage and transmission of movies using currently available storage media (notably DVDs) and transmission bandwidth. This container enforces the video codec, you can only define quality parameters, and the audio codec.
File Extensions: .dvd, .vob, .mpg, .mpeg

MPEG-4::

While being a [video codec](#), it is also a real container, in which you can store video and audio streams using various codecs. It is widely supported by many modern software and hardware players.
File Extensions: .mp4, .mpg, .mpeg

AVI::

A derivative of the Resource Interchange File Format (RIFF). One of the first and most widely used video container format.
File Extension: .avi

QuickTime::

A multi-tracks format. QuickTime and MP4 container formats can use the same codecs. They are mostly interchangeable in a QuickTime-only environment. MP4, being an international standard, has more support.
File Extension: .mov

DV::

An intra-frame video compression scheme, used by many digital camcorders back in the days. It uses the discrete cosine transform (DCT, similar algorithm to JPEG) to compress video on a frame-by-frame basis. Audio is stored uncompressed. This container enforces the video codec, you can only define quality parameters.

File extension: .dv

Ogg::

A free open-standard container format, that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.

File Extensions: .ogg, .ogv

Matroska::

A free open-standard container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.

File Extension: .mkv

Flash::

A container file format used to deliver video over the Internet using Adobe Flash Player. This container enforces the video codec, you can only define quality parameters.

File Extension: .flv

WebM::

A free open-standard container format, designed to be used for internet streaming. Note that this container can only hold a VP9 video codec, and Vorbis or Opus audio codecs.

File Extension: .webm

FFmpeg Video Codecs

These options are not available with all [Containers](#).

No Video::

For audio-only encoding.

DNxHD::

Intended to be usable as both an intermediate format suitable for use while editing, and as a presentation format. It can be either lossless or lossy.

<u>DV::</u>	See <u>Containers</u> .
<u>FFmpeg video codec #1::</u>	FFV1 is a lossless intra-frame video codec. It can use either variable length coding or arithmetic coding for entropy coding. The encoder and decoder are part of the free, open-source library libavcodec in FFmpeg. Supports an alpha channel.
<u>Flash Video::</u>	See <u>Containers</u> .
<u>H.264::</u>	A modern variation of the MPEG-4 family, this lossy codec is very commonly used. It offers a very good compression/quality ratio.
<u>HuffYUV::</u>	Lossless video codec created by Ben Rudiak-Gould which is meant to replace uncompressed YCbCr as a video capture format.
<u>MPEG-1::</u>	See <u>Containers</u> .
<u>MPEG-2::</u>	See <u>Containers</u> .
<u>MPEG-4(DivX)::</u>	Inherits many of the features of MPEG-1, MPEG-2 and other related standards, but also adds new features.
<u>PNG::</u>	Lossless, this stores each frame as an independent image in the video stream. Compression will be poor, but as every frame is fully self-contained, scrubbing and editing can be simpler. Supports an alpha channel.
<u>QuickTime Animation::</u>	Original format of QuickTime videos. Supports an alpha channel.
<u>Theora::</u>	A free open-standard lossy codec designed together with the <u>Ogg container</u> .
<u>WEBM / VP9::</u>	A free open-standard lossy video compression format. One of the most recent codecs, it is widely used for internet streaming.
<u>AV1::</u>	A free open-standard lossy video compression format, designed as a successor to <i>VP9</i> . AV1 offers great compression rates and visual quality, <i>AV1</i> produces video files that are about 30% more space efficient than <i>VP9</i>

FFmpeg Audio Codecs

No Audio::	For video-only encoding.
<u>AAC::</u>	Advanced Audio Codec, a standardized, lossy compression and encoding scheme for digital audio. AAC generally achieves better sound quality than MP3 at similar bit rates.
<u>AC3::</u>	Audio Codec 3, an audio compression technology developed by Dolby Laboratories.
<u>FLAC::</u>	Free Lossless Audio Codec. Digital audio compressed by FLAC's algorithm can typically be reduced to 50-60% of its original size.
<u>MP2::</u>	A lossy audio compression format.
<u>MP3::</u>	A lossy audio compression format, widely used as final audio format.
<u>Opus::</u>	A lossy audio compression format, designed to encode speech or general audio and is intended to replace the <i>Vorbis</i> codec.
<u>PCM::</u>	Pulse Code Modulation, a method used to digitally represent sampled analog signals. It is the standard form for digital audio in computers and various Blu-ray, Compact Disc and DVD formats, as well as other uses such as digital telephone systems.
<u>Vorbis::</u>	An open-standard, highly-compressed format comparable to MP3 or AAC. Vorbis generally achieves better sound quality than MP3 at similar bit rates.

Known Limitations

Video Output Size

Some codecs impose limitations on output size, H.264, for example requires both the height and width to be divisible by 2.

Importing & Exporting Files

Reference

Menu:: *Topbar ▶ File ▶ Import/Export*

Sometimes you may want to utilize files that either came from other 2D or 3D software, or you may want to use the things you have made in Blender and edit them in other software. Luckily, Blender offers a wide range of file formats (e.g. ABC, USD, OBJ, FBX, PLY, STL, etc.) that can be used to import and export.

Popular formats are enabled by default, other formats are also supported and distributed with Blender, these can be enabled in the Preferences through the use of [Add-ons](#).

- [Alembic](#)
- [Collada \(Legacy\)](#)
- [Universal Scene Description](#)
- [Wavefront OBJ](#)
- [Stanford PLY](#)
- [STL](#)
- [Import/Export SVG as Grease Pencil](#)
- [Export Grease Pencil as \(PDF\)](#)

See also

More information on the add-ons to import/export these file types can be found in the [add-ons section](#).

Alembic

From the [Alembic home page](#):

Alembic is an open computer graphics interchange framework. Alembic distills complex, animated scenes into a non-procedural, application-independent set of baked geometric results. This ‘distillation’ of scenes into baked geometry is exactly analogous to the distillation of lighting and rendering scenes into rendered image data.

Alembic is focused on efficiently storing the computed results of complex procedural geometric constructions. It is very specifically **not** concerned with storing the complex dependency graph of procedural tools used to create the computed results. For example, Alembic will efficiently store the animated vertex positions and animated transforms that result from an arbitrarily complex animation and simulation process which could involve enveloping, corrective shapes, volume-preserving simulations, cloth and flesh simulations, and so on. Alembic will not attempt to store a representation of the network of computations (rigs, basically) which are required to produce the final, animated vertex positions and animated transforms.

In brief, Alembic can be used to write an animated mesh to a drive, and read it back quickly and efficiently. This means that a mesh can be animated with a very CPU-intensive rig and then ‘baked’ to an Alembic file. Finally it can be load into the shot file for shading and lighting with only moderate CPU usage.

Due to the Open Source nature of the Alembic standard as well as the C++ library implementing that standard, **Blender can be used in a hybrid pipeline**. For example, other software, such as Houdini or Maya, can export files to Alembic, which can then be loaded, shaded, and rendered in Blender. It is also possible to animate characters (or other models) in Blender, export to Alembic, and load those files into other software for further processing.

Importing Alembic Files

When importing an Alembic file, [Mesh Sequence Cache modifiers](#) are automatically added to time-varying meshes. For time-varying object transforms (so animation of rotation, location, or scale) the [Transform Cache Constraint](#) is used.

General

Scale

Value by which to enlarge or shrink the objects with respect to the world's origin

Options

Relative Path

Select the file relative to the blend-file.

Set Frame Range

If checked, update scene's start and end frame to match those of the Alembic archive.

Is Sequence

Set to true if the cache is split into separate files.

Validate Meshes

Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Always Add Cache Reader

Add cache modifiers and constraints to imported objects even if they are not animated so that they can be updated when reloading the Alembic archive.

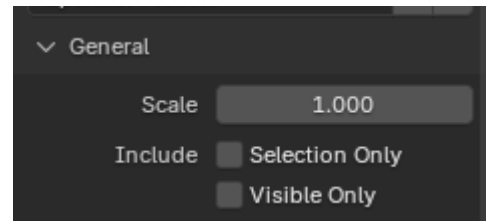
Exporting to Alembic Files

This section describes the effect of the different export options.

General

Scale

This sets the global scale of the Alembic file. Keep it at the default value of 1.0 to use Blender's units.



Include

Selected Objects

When enabled, exports only the selected objects. When disabled, all objects are exported.

General options.

Visible Objects

Limits the export to scene collections that are currently visible.

Scene

Frame Start, End

Sets the frame range to export to Alembic. This defaults to the current scene frame range.

Sub-frame Sampling

These options control the sub-frame sampling of animations.

Samples Transform

Transform Samples sets the number of times per frame at which animated transformations are sampled and written to Alembic.

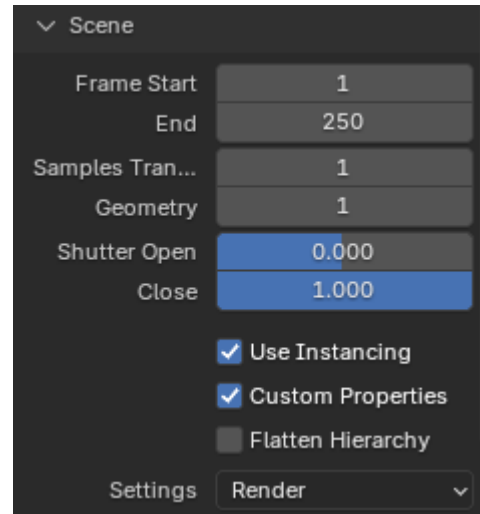
Geometry

Geometry Samples sets the same, but then for animated geometry.

Shutter Open, Close

Shutter Open/Close define the interval [open, close] over which those samples are taken. The valid range is -1 to 1, where -1 indicates the previous frame, 0 indicates the current frame, and 1 indicates the next frame.

For example, if information for detailed mesh motion blur is desired, some subframes around the current frame can be written to Alembic by using a sample count of 5, Shutter Open at -0.25 and Shutter Close at 0.25. This mimics a “180 degree” shutter, opening 90 degrees before the frame and closing 90 degrees after the frame.



Scene options.

Use Instancing

Exports data of [duplicated](#) or [instanced](#) objects as Alembic instances; speeds up the export and can be disabled for compatibility with other software.

Custom Properties

When enabled (which it is by default), custom properties are exported to Alembic as well. The following custom property types are supported:

- Numbers (int, float) and strings. These are exported as arrays of a single element, so 47 will be exported as [47] to Alembic, and "Agent" to ["Agent"]. This matches the behavior of many other DCCs.
- Lists of numbers and strings. These are exported as-is, so [327, 47] is exported as [327, 47].
- Matrices and nested arrays of numbers. These are flattened into one long list, so a 3×2 matrix of numbers will become a list of 6 numbers. Similarly, nested lists [[1, 2, 3], [4, 5], [6]] will be exported as [1, 2, 3, 4, 5, 6].
- Numbers can be animated as well.

Flatten Hierarchy

When disabled, parent/child relations between objects are exported too. Any parent object that is not exported itself, but with children that *are* exported, is replaced by an empty. When enabled, parent/child relations are not exported, and transformations are all written in world coordinates.

Settings

Determines visibility of objects, modifier settings, and other areas where there are different settings for viewport and rendering.

Render:: Use Render settings for object visibility, modifier settings, etc.

Viewport:: Use Viewport settings for object visibility, modifier settings, etc.

Geometry

UV Coordinates

When enabled, UV maps are exported. Although the Alembic standard only supports a single UV map, Blender exports all UV maps in a way that should be readable by other software.

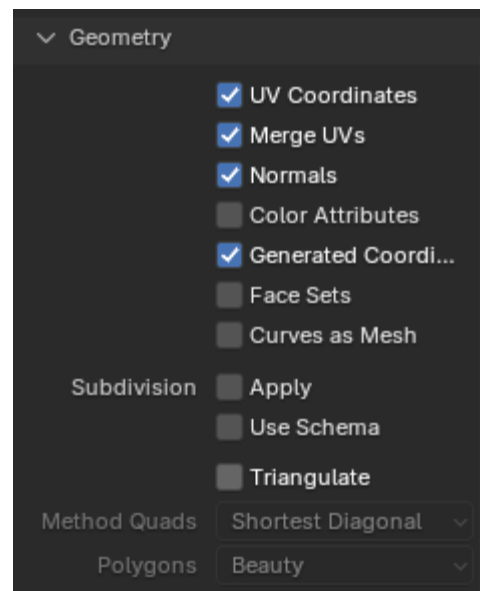
Merge UVs

When enabled, UVs sharing the same vertex and location will be merged into a single UV. The exported file can be slightly smaller.

Normals

When enabled, an object's [Normals](#) are exported. See [Custom Split Normals of Meshes](#) below for more information.

Color Attributes



Geometry options.

When enabled, exports Color Attributes.

Face Sets

Exports the material names per face. The material data is not exported but only material names.

Subdivisions

Apply

Applies any [Subdivision Surface modifiers](#) before writing to Alembic.

Use Schema

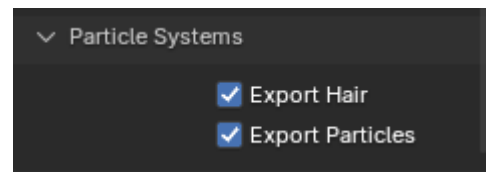
Writes polygonal meshes using the “SubD” Alembic schema, rather than the “PolyMesh” schema. This sets an import option for the program, with which the file is opened, to apply its form of a non-destructive subdivision.

Triangulate

Triangulates the mesh before writing to Alembic. For more detail on the specific option see the [Triangulate modifier](#).

Particle Systems

Alembic has no support for Particle Systems, in the same way that it does not support armatures.



Export Hair

Hair is exported as animated zero-width curves.

Particle Systems options.

Export Particles

Particles are exported as animated points.

Custom Split Normals of Meshes

Blender supports the import and export of [custom normals](#) to Alembic files. As a basic rule of thumb, a completely smooth mesh will be exported without normals and thus produce the smallest Alembic file. This is reflected in the importer; an Alembic mesh without normals is loaded as a smooth mesh.

On export, for every mesh:

- If it has *Custom Loop Normals* then the loop normals are exported.
- If one or more polygons are marked flat then loop normals are also exported.
- Otherwise, no normals are exported.

On import, when the Alembic mesh contains:

- Loop normals (kFacevaryingScope) are used as custom loop normals.
- Vertex normals (kVertexScope or kVaryingScope) are convert to loop normals, and handle as above.
- If there are no normals then the mesh is marked as smooth.
- Unsupported normal types (kConstantScope, kUniformScope, kUnknownScope) are handled as *no normals*.

When an imported mesh does not contain normals, the final look can be controlled using the [Normal's Shading](#).

Handling Time

Unlike Blender and many other applications and file formats, Alembic files don't have any concept of frames. Alembic works purely with time, and values that are sampled over time. For example, there is no way to distinguish 30 FPS with 2 samples per frame, and 60 FPS with 1 sample per frame. This has caused many developers to just [hard-coded 24 FPS](#) when reading Alembic files.

Blender uses the current scene frame rate to convert a frame number (in Blender) to a time in seconds (in Alembic). As a result, you can import an

Alembic file that was produced at 120 FPS into a Blender scene that is 30 FPS and still not see any time stretching.

Collada (Legacy)

Important

COLLADA I/O support is now considered as a legacy feature in Blender, and will be removed in a future release. Please see the [official announcement](#) for more insight on this topic.

The COLLADA™ module has been implemented as a flexible tool for exporting and importing .dae files. A design goal is to provide a set of parameters which should make it possible to export/import Collada files from/to a variety of tools. But please be aware that the Collada module is still a work in progress. So it may be possible that your particular usage scenario is not yet supported.

Collada Exporter

Reference

Menu:: *File ▶ Export Collada (.dae) (Legacy)*

Operator Presets

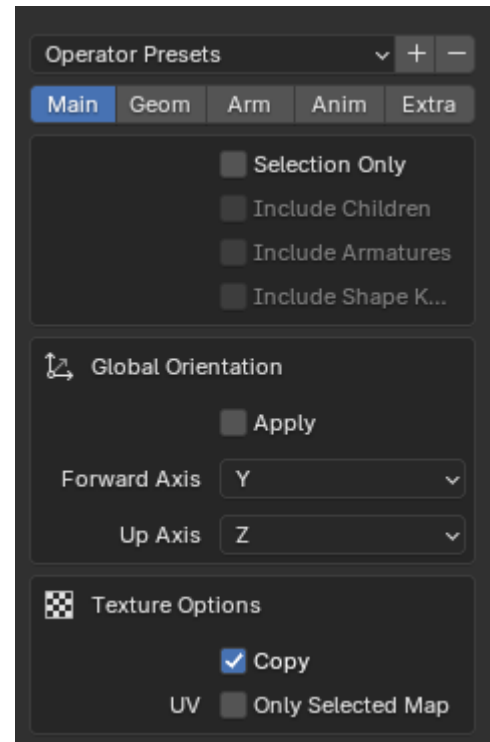
There are two operator presets (see top of the Sidebar) for Second Life (SL) users:

- *Second Life Static* – is good for exporting static meshes.
- *Second Life Rigged* – is good for exporting the SL default character.

Note

Special Notes for Second Life users:

- Please use the Operator presets. All other export settings will not work for Second Life.
- The character orientation needs to be such that the character looks towards positive X.
- Scale and Rotation must be applied before the export!



Main

Selection Only

When *Selection Only* is enabled, then only the selected objects will be exported. Otherwise the entire scene is exported with all visible and all invisible objects.

Include Children

When this option is enabled then all children of the selected objects will also be exported regardless of their selection state.

Hint

You can select only an armature, then using this option, all rigged meshes attached to the armature will also be exported.

Include Armatures

When this option is enabled, then all armatures related to the selected objects will also be exported regardless of their selection state.

Hint

You can select only the objects, then in the exporter enable this option to export the armature data also.

Include Shape Keys

Includes the application of shape keys by exporting meshes with the current shape key configuration baked in.

Global Orientation

Apply

Rotate all root objects to match the global orientation settings otherwise set the global orientation per Collada asset.

Forward / Up Axis

Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, it is common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Texture Options

Copy

When you export images either material based image textures, then the exporter creates absolute file references in the export file.

But if the *Copy* option is enabled, the exporter will create copies of the images instead and place the copies besides the export file. In that case the file references are made relative.

Only Selected UV Map

When your mesh contains multiple UV layers, then Blender exports all layers by default. This option allows you to only export the active (selected) UV layer.

Geometry

Export Data Options

Triangulate

The mesh can be triangulated on-the-fly. The triangulation is based on the same function which is used in the *Triangulate Faces* tool for triangulating the current selection of faces. For full control over the triangulation you can do this manually before exporting. However, this option allows to apply the triangulation only on the exported data; the mesh itself is not affected.

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

Resolution

Controls whether to apply the 3D Viewport resolution or the render resolution for modifiers that provide a preview mode and a render mode.

Transform

Collada supports two types of transformation matrix specifications. Either as <Matrix> or as a set of transformation decompositions (for move, rotate and scale). Note that the exporter will not strictly follow this option setting, but will rather take it as a hint to use the option if ever possible. This is so because some of the exported data types have specific rules about how the transformation matrix has to be exported. This is ongoing development and a less ambiguous method may be provided in the future.

Armature

Armature Options

Deform Bones Only

When this option is enabled, then the exporter strips all non-deforming bones from the exported armatures. This option is useful when your armatures contain control bones which are not actually part of the character skeleton. For example you can export the Avastar rig with this option enabled. The resulting exported rig is compatible with Second Life. But please note the restrictions further below.

Export to SL/OpenSim

When this option is enabled, some issues with bone orientation are calculated differently and is designed to be used to export to Second Life or OpenSim.

This is only relevant for rigged meshes, for static meshes it just does nothing at all.

Animation

Extra

Collada Options

Use Object Instances

In Blender you can reuse the same mesh for multiple objects. This is named “object instantiation”. When you enable this option, then Blender will propagate object instantiation to the Collada file.

Use Blender Profile

Collada can be extended with tool specific data (profiles). Blender has its own (unofficial) profile that allows to export rig information into the Collada file. Later It can be used to reconstruct the rig when it should ever be necessary to import a dae file back into Blender.

Sort by Object Name

The export order of data is bound to internal object order and it can not be influenced in a reliable way. This option ensures that the Geometry nodes and the Object nodes are both exported in alphabetical order.

Keep Bind Info

When a rig is imported to Blender, the rig's bind pose will be used as Blender's rest pose. So all Matrix information of the original rest pose is lost. But in some cases you may want to preserve the original rig information. This option checks each bone for having two arrays:

- `rest_mat` – an array of 16 floats which represent the bone's original rest-pose matrix.
- `bind_mat` – an array of 16 floats which represent the bone's original bind-pose matrix.

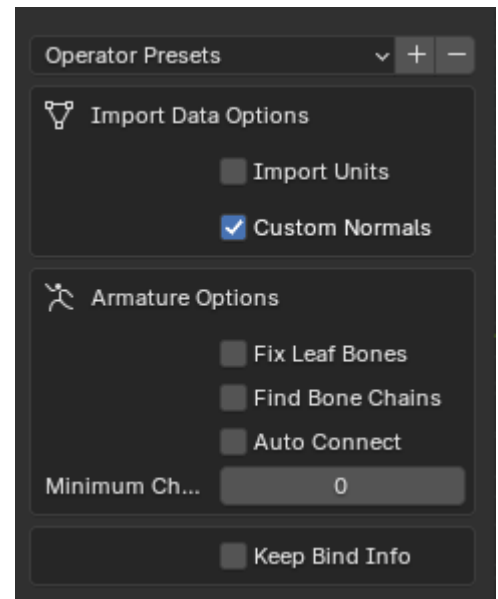
If the arrays are present, then those arrays will be used instead of the current rest pose/bind pose. Those two arrays are either created by a previous Collada import (see [Collada Importer](#) below), or they can be created manually, or by an add-on (script based).

Collada Importer

Reference

Menu:: *File ▶ Export Collada (.dae) (Legacy)*

The Collada importer is mostly driven by the imported data. There is one option for controlling the import units:



Import Data Options

Custom Normals

Use the mesh normals defined in the collada file, if they are defined, otherwise Blender will recompute them during the import process.

Import Units

If not enabled the imported data will be rescaled according to the currently used unit system. If this option is enabled, then Blender will adjust itself to the unit system as provided by the Collada file.

Armature Options

Fix Leaf Bones

Collada only records “joints” which is mostly similar to Blender’s bone heads. But when you import a Collada file then the bone head/tail are not defined. This does not matter for connected bones where the bone parent only has one child. In that case the parent bone’s end location is adjusted to the child’s joint position. But especially for unconnected bones and for bones with more than one child a problem arises.

When the *Fix Leaf Bones* option is enabled then Blender tries to guess where the bone head/tail of unconnected bones would best be placed. If

the option is disabled, then the bone head/tail are placed at an offset along the Y axis. That is why bones often point towards the Y axis.

Find Bone Chains

When a bone has multiple children, then it is not defined which (if any) of the children should be connected to the bone. When the *Find Bone Chains* option is enabled, then Blender determines the longest bone chain (of children) for each bone. All bones along this chain will then be auto connected.

If the option is disabled, then children will only be connected to parents, if the parent has only one child. But see the *Auto Connect* option below.

Auto Connect

When this option is enabled, then children will automatically be connected to their parents, if the parent has only one child.

Keep Bind Info

When this option is enabled, then the importer creates two custom properties for each bone:

- `rest_mat` – an array of 16 floats which represent the bone's original rest-pose matrix.
- `bind_mat` – an array of 16 floats which represent the bone's original bind-pose matrix.

Those two arrays can later be used when you want to export the rig again and be sure the original rest pose/bind pose combination must be used.

Technical Details

Mesh

Import

Supported geometry types are:

- Tris (not tested)
- Polylist
- Polygons
- N-gons
- Tri-fans (not tested)
- Lines

Export

Mesh data is exported as <polylist>, <lines> and <vertices>.

Light

Import

Blender does a best effort on importing lights from a dae-file. If a Blender profile is detected for lights, all values from these will be used instead. This ensures full re-import from a Blender exported dae-file. <extra> support has been added in Blender 2.57.

Export

A Blender profile for lights has been added through the <extra> tag. The entire Light struct from Blender will be exported through this profile, with the exception of light curve falloff.

Animation

Export & Import

- Support for object (mesh, camera, light) transform animations. Only Euler rotations, which is the default option for Objects, can be exported. For armature bone animations, Euler and quaternion rotation types are supported.

- Import and export of animations for the following parameters are supported:
 - Light
 - Camera
 - Material effects
- Non-skin controlling armature bone animation.
- Animations of armatures with skin deforming bones.
- Animations of armatures in Object Mode.
- Fully rigified armature animations (referring to the Rigify add-on). For export of rigified armature animations:
 - Run the [Bake Action](#) operator.
 - If you have only the deform bones selected check *Only Selected*. This will give smaller dae. Otherwise uncheck *Only Selected*.
 - Check *Clear Constraints*.
 - Bake Action.
 - Select the mesh and the deform bones. Then export to Collada while checking only selected option. (Selecting only the Mesh and bones is not strictly necessary. Selecting and export only selected will give smaller dae.)
 - [Demonstration video](#)

For bone nodes which are leaf nodes in the armature tree, or if a bone has more than one child, a Blender profile for tip with an <extra> tag, is added for those joint nodes. To correctly derive the bone-to-tail location on re-import.

Note

Important Things to Remember

- Object and data-block names are constrained to 21 characters (bytes).
- UV layer names are constrained to 32 characters (bytes).
- Only armature animation on mesh, single skin controller.
- No support for modifiers yet.

When importing a dae-file that has <instance_node> on exporting this information is essentially lost and these nodes will be <node>s.

Universal Scene Description

Importing USD Files

[USD](#) files typically represent the scene as a hierarchy of primitives, or [prims](#). Individual prims contain data to describe scene entities, such as geometry, lights, cameras and transform hierarchies. Blender's USD importer converts USD prims to a hierarchy of Blender objects. Like the USD exporter, the importer does not yet handle more advanced USD concepts, such as layers and references.

The following USD data types can be imported as Blender objects:

- Cameras
- Curves
- Lights
- Materials
- Meshes
- Primitive Shapes
- Volume

For more information on how the various data types are handled, see the following descriptions of the [Import Options](#).

Note

When importing a [USDZ archive](#), it is important to carefully consider the [Import Textures](#) option to determine whether and how to copy texture files from the zip archive.

Xform and Scope Primitives

USD provides an Xform prim type, containing transform data, which can be used to represent transform hierarchies and to organize the scene. Such Xform prims are imported as Blender empty objects.

USD also supports Scope primitives, which are entities that do not contain transform data, but which serve to group other element of the scene. Blender doesn't have an exact counterpart to the concept of a scope, so such primitives are imported as Blender empties located at the origin. This is an imperfect representation, because empty objects have a transform and Scopes do not, but this approach nonetheless helps preserve the structure of the scene hierarchy.

PointInstancer Primitives

USD provides a UsdGeomPointInstancer prim type, containing instances that are scattered on a primitive's points.

These are imported into Blender as Point Clouds using a [Geometry Nodes Modifier](#) and the [Instance on Points Node](#).

Animations

The importer supports two types of animation:

- **Animating transforms:** If a USD primitive has time-varying transform data, a [Transform Cache](#) constraint will be added to the imported Blender object.
- **Animating geometry:** Animating mesh and curve geometry is supported by adding a [Mesh Sequence Cache](#) modifier to the imported data. Geometry attribute ([USD Primvar](#)) animation is currently supported only for Color Attributes and UVs. Note that USD file sequences (i.e. a unique file per frame) are not yet supported.

Materials

If a USD mesh or geometry subset has a bound material, the importer will assign to the Blender object a material with the same name as the USD

material. If a Blender material with the same name already exists in the scene, the existing material may be used, depending on the [Material Name Collision](#) option. Otherwise, a new material will be created.

If the USD material has a [USD Preview Surface](#) shader source, the [Viewport Display](#) color, metallic, and roughness are set to the corresponding USD Preview Surface input values.

There is also an *Import USD Preview* option to convert USD Preview Surface shaders to Blender [Principled BSDF](#) shader nodes. This option can be lossy, as it does not yet handle converting all shader settings and types, but it can generate approximate visualizations of the materials.

Coordinate System Orientation

If the imported USD is Y up, a rotation will be automatically applied to root objects to convert to Blender's Z up orientation.

Import Options

The following options are available when importing from USD:

General

Path Mask

Import only the subset of the USD scene rooted at the given primitive.

Include

Visible Primitives Only

Do not import invisible USD primitives. Only applies to primitives with a non-animated [visibility](#) attribute. Primitives with animated visibility will always be imported.

Defined Primitives Only

When disabled this allows importing USD primitives which are not defined, such as those with an override specifier.

Set Frame Range

Update the scene's start and end frame to match those of the USD stage.

Create Collection

Add all imported objects to a new collection.

Relative Path

Select the file relative to the blend-file.

Scale

Value by which to scale the imported objects in relation to the world's origin.

Light Intensity Scale

Scale for the intensity of imported lights.

Custom Properties

Behavior when importing USD attributes as [Custom Properties](#).

None::	Does not import USD custom attributes.
User::	Imports USD attributes in the <code>userProperties</code> namespace as custom properties. The namespace will be stripped from the property names.
All Custom::	Imports all USD custom attributes as custom properties. Namespaces will be retained in the property names.

Object Types

Cameras

Import cameras (perspective and orthographic).

Curves

Import curve primitives, including USD basis and NURBS curves.
(Note that support for Bézier basis is not yet fully implemented.)

Lights

Import lights. Does not currently include USD dome, cylinder or geometry lights.

Materials

Import materials.

Meshes

Import meshes.

Volumes

Import USD OpenVDB field assets.

Point Clouds

Imports USD UsdGeomPoints as a [Point Cloud](#) object.

USD Shapes

Imports USD primitive shapes (cubes, spheres, cones, ect) as Blender meshes.

Display Purpose

Render

Include primitives with purpose render.

Proxy

Include primitives with purpose proxy.

Guide

Include primitives with [purpose](#) guide.

Material Purpose

Attempt to import materials with the given purpose. If no material with this purpose is bound to the primitive, fall back on loading any other bound material.

All Purpose::

Attempt to import allPurpose materials.

Preview::

Attempt to import preview materials. Load allPurpose materials as a fallback.

Full::

Attempt to import full materials. Load allPurpose or preview materials, in that order, as a fallback”.

Geometry

UV Coordinates

Read mesh UV coordinates.

Color Attributes

Convert the USD mesh displayColor values to Blender’s Color Attributes.

Mesh Attributes

Read USD Primvars as mesh attributes.

Subdivision

Create Subdivision Surface modifiers based on the USD SubdivisionScheme attribute.

Validate Meshes

Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Rigging

Shape Keys

Imports USD blend shapes as Blender’s [Shape Keys](#).

Armatures

Imports USD skeletons as Blender’s [Armatures](#).

Materials

Import All Materials

Also import materials that are not used by any geometry. Note, when this option is false, materials referenced by geometry will still be imported.

Import USD Preview

Convert USD Preview Surface shaders to Principled BSDF shader networks.

Create World Material

Converts the first discovered USD dome light to a [world background shader](#).

Set Material Blend

If the *Import USD Preview* option is enabled, the material blend method will automatically be set based on the opacity and opacityThreshold shader inputs, allowing for visualization of transparent objects.

Material Name Collision

Behavior when the name of an imported material conflicts with an existing material.

Make Unique:: Import each USD material as a unique Blender material.

Reference Existing:: If a material with the same name already exists, reference that instead of importing.

Textures

When importing a USDZ package, the following options specify whether and how texture asset dependencies of the USD should be copied from the zip archive so they can be loaded into Blender.

Import Textures

Behavior when importing textures from a USDZ archive.

None::	Don't import textures. Note that, with this option, material textures may fail to be resolved in Blender.
Packed::	Import textures as packed data in the Blender file.
Copy::	Copy files to the directory specified in the Textures Directory option.

Textures Directory

Path to the directory where imported textures will be copied, when the **Import Textures** mode is **Copy**.

Note that the default textures directory is the relative path `//textures`, which requires the Blender file to have been saved before importing, so the relative path can be resolved.

File Name Collision

Behavior when the name of an imported texture file conflicts with an existing file.

Use Existing::	If a file with the same name already exists, use that instead of copying.
Overwrite::	Overwrite existing files.

Particles and Instancing

Scene Instancing

Import USD scene graph instances as collection instances, otherwise they are imported as copies.

Exporting to USD Files

Universal Scene Description (USD) files can contain complex layering, overriding, and references to other files. Blender's USD Exporter takes a much simpler approach. When exporting, all visible, supported objects in the scene are exported, optionally limited by their selection state. Blender does not (yet) support exporting invisible objects, USD layers, variants, etc.

The following objects can be exported to USD:

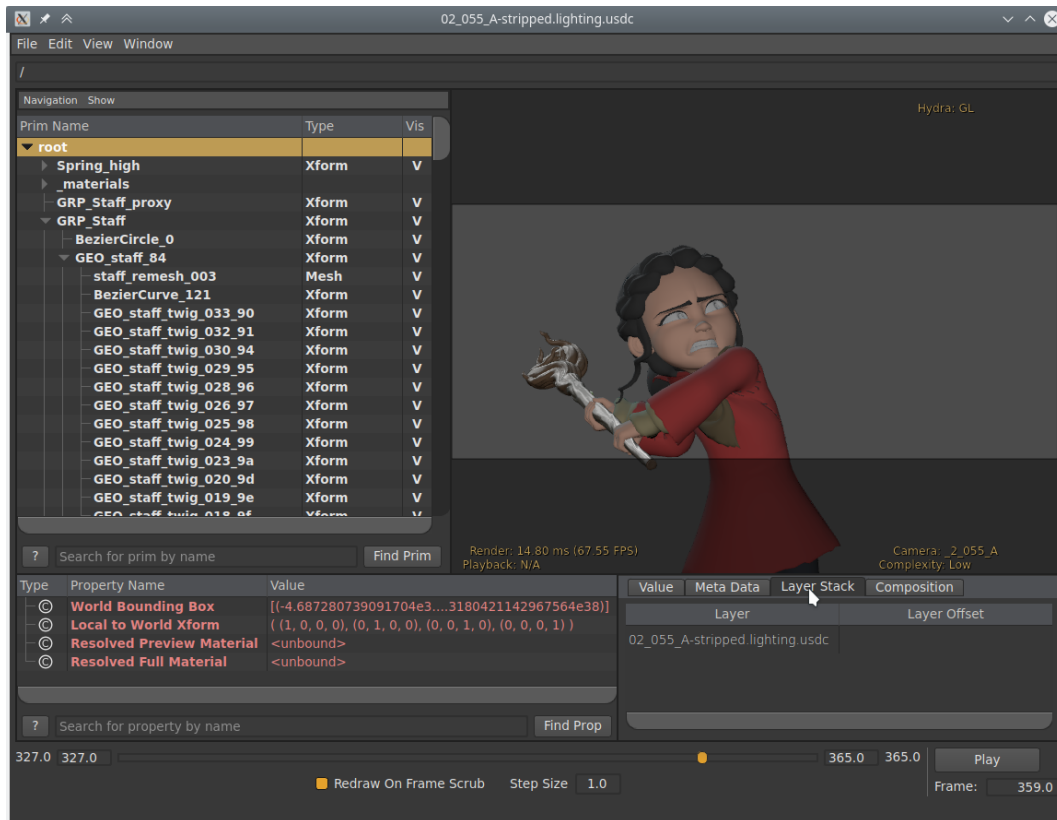
- Meshes (of different kinds, see below).
- Cameras (perspective cameras only at the moment, not orthogonal ones).
- Light (all types except area lights).
- Hair (exported as curves, and limited to parent strands).
- Volume (both static and animated volumes).
- Armatures

When exporting an animation, the final, evaluated mesh is written to USD. This means that the following meshes can be exported:

- Static meshes.
- Deforming meshes; here the topology of the mesh does not change, but the locations of the vertices change over time. Examples are animated characters or bouncing (but not cracking) objects.
- Arbitrarily animated meshes; here the topology does change. An example is the result of a fluid simulation, where splashes of fluid can break off the main body.
- Metaballs are exported as animated meshes.

Note

To export the Blender scene as a [USDZ archive](#), set the file extension of the output file to .usdz. The exported USDZ package will be a zip archive containing the USD and its texture file dependencies.



Shot from [Spring](#) exported to USD and opened in USDView.

Export Options

The following options are available when exporting to USD:

General

Root Prim

If set, add a transform primitive with the given path to the stage as the parent of all exported data.

Include

Selection Only

When checked, only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered 'selected' when their instancer is selected.

Visible Only

Only exports objects that are not [hidden](#). Invisible parents of exported objects are exported as empty transforms.

Animation

When checked, the entire scene frame range is exported. When unchecked, only the current scene frame is exported.

Blender Data

Custom Properties

Exports [Custom Properties](#) as USD attributes. The *Namespace* property is used to determine the namespace that the attributes are written to.

Namespace

If set, add the given namespace as a prefix to exported custom property names. This only applies to property names that do not already have a prefix (e.g., it would apply to name bar but not foo:bar) and does not apply to Blender object and data names which are always exported in the `userProperties:blender` namespace.

By default, `userProperties` namespace is used.

Blender Names

Author USD custom attributes containing the original Blender object and object data names.

Allow Unicode

Preserves UTF-8 encoded characters when writing USD prim and property names (requires software utilizing USD 24.03 or greater when opening the resulting files).

File References

Relative Paths

Use relative paths to reference external files (i.e. textures, volumes) in the exported USD file, otherwise use absolute paths.

Convert Orientation

Convert orientation axis to a different convention to match other applications. Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Forward / Up Axis

By mapping these to different axes you can convert rotations between applications default up and forward axes.

Xform Ops

The type of transform operators to use to transform prims.

Translate, Rotate, Scale::

Export with translate, rotate, and scale Xform operators.

Translate, Orient, Scale::

Export with translate, orient quaternion, and scale Xform operators.

Matrix::

Export matrix operator.

Use Settings for

Determines the whether to use *Viewport* or *Render* visibility of collection, modifiers, or any other property that can be set for both the *Viewport* and *Render*.

Object Types

Meshes

Exports [Mesh Objects](#)

Lights

Exports [Light Objects](#) USD does not directly support spot lights, so those are not exported.

Cameras

Exports [Camera Objects](#) Only perspective cameras are exported.

Curves

Exports [Curve Objects](#)

Point Clouds

Exports [Point Cloud Objects](#) as UsdGeomPoints primitives.

Volumes

Exports [Volume Objects](#)

Hair

Exports parent hair strands are exported as a curve system. Hair strand colors are not exported.

Geometry

UV Maps

When checked, includes UV coordinates for exported meshes. The name of the UV map in USD is the same as the name in Blender.

Rename UV Maps

Exports UV maps using the USD default name (st) as opposed to Blender's default name (UVMap).

Normals

When checked, includes normals for exported meshes. This includes custom loop normals.

Triangulate

Triangulates the mesh before writing. For more detail on the specific option see the [Triangulate modifier](#).

Rigging

Shape Keys

Export shape keys as USD blend shapes.

Absolute shape keys are not supported.

Armatures

Export [Armatures](#) and meshes with [Armature Modifiers](#) as USD skeletons and skinned meshes.

Limitations:

- Modifiers in addition to Armature modifiers will not be applied.
- Bendy bones are not supported.

Only Deform Bones

Only export [deform bones](#) and their parents.

Materials

Exports material information of the object. By default the exporter approximates the [Principled BSDF](#) node tree by converting it to USD's Preview Surface format.

When a mesh has multiple materials assigned, a geometry subset is created for each material. The first material (if any) is always applied to the mesh itself as well (regardless of the existence of geometry subsets), because the Hydra viewport does not support materials on subsets. See [USD issue #542](#) for more information.

Note

If *USD Preview Surface Network* and *MaterialX Network* are disabled, the material is set to the viewport materials of meshes.

USD Preview Surface Network

Approximates a [Principled BSDF](#) node tree to by converting it to USD's Preview Surface format.

Warning

Not all nodes are supported; currently only Diffuse, Principle, Image Textures, and UVMap nodes are support.

MaterialX Network

Generates material shading graphs using the [MaterialX](#) standard. This standard is designed to support a high amount of interoperability among DCCs <Digital Content Creation>. In Blender, MaterialX supports most of the shader nodes and their functionality but has a few caveats (see below).

Implementation Caveats

When using the Principled BSDF, the resulting graph is very usable. However, when using some of the other BSDFs, some of the generated shading graphs are difficult for other DCC's to understand.

Convert World Material

Convert the [world material](#) to a USD dome light. Currently works for simple materials, consisting of an environment texture connected to a background shader, with an optional vector multiply of the texture color.

Export Textures

Method for exporting textures.

Keep::	Use original location of textures.
Preserve::	Preserve file paths of textures from already imported USD files. Export remaining textures to a 'textures' folder next to the USD file.
New Path::	Export textures to a 'textures' folder next to the USD file.

Overwrite Textures

Allow overwriting existing texture files when exporting textures.

USDZ Texture Downsampling

Choose a maximum size for all exported textures.

Keep::	Keep all current texture sizes.
256::	Resize to a maximum of 256 pixels.
512::	Resize to a maximum of 512 pixels.
1024::	Resize to a maximum of 1024 pixels.
2048::	Resize to a maximum of 2048 pixels.
4096::	Resize to a maximum of 4096 pixels.
Custom::	Specify a custom size. USDZ Custom Downscale Size The size in pixels of the <i>Custom</i> downsampling.

Experimental

Instancing

As this is an experimental option. When unchecked, duplicated objects are exported as real objects, so a particle system with 100 particles that is displayed with 100 meshes will have 100 individual meshes in the exported file. When checked, duplicated objects are exported as a reference to the original object. If the original object is not part of the export, the first duplicate is exported as real object and used as reference.

Exporter Limitations

Single-sided and Double-sided Meshes

USD seems to support neither per-material nor per-face-group double-sidedness, so Blender uses the flag from the first material to mark the entire mesh as single/double-sided. If there is no material it defaults to double-sided.

Mesh Normals

The mesh subdivision scheme in USD is ‘Catmull-Clark’ by default, but Blender uses ‘None’ instead, indicating that a polygonal mesh is

exported. This is necessary for USD to understand the custom normals; otherwise the mesh is always rendered smooth.

Vertex Velocities

Currently only fluid simulations (not meshes in general) have explicit vertex velocities. This is the most important case for exporting velocities, though, as the baked mesh changes topology all the time, and thus computing the velocities at import time in a post-processing step is hard.

Materials

When there are multiple materials, the mesh faces are stored as geometry subset and each material is assigned to the appropriate subset. If there is only one material this is skipped. Note that the geometry subsets are not time-sampled, so it may break when an animated mesh changes topology.

Hair

Only the parent strands are exported, and only with a constant color. No UV coordinates, and no information about the normals.

Camera

Only perspective cameras are exported.

Lights

USD does not directly support spot lights, so those are not exported.

Particles

Particles are only written when they are alive, which means that they are always visible. There is currently no code that deals with marking them as invisible outside their lifespan.

Objects instanced by particle system are exported by suffixing the object name with the particle's persistent ID, giving each particle transform a unique name.

Instancing/Referencing

This is still an experimental feature that can be enabled when exporting to USD. When enabled, instanced object meshes are written to USD as references to the original mesh. The first copy of the mesh is written for real, and the following copies are referencing the first. Which mesh is considered 'the first' is chosen more or less arbitrarily.

USDZ

Due to a current limitation in the USD library, UDIM textures cannot be included in the USDZ archive. This limitation will likely be addressed in a future version of USD. (See [USD pull request #2133](#).)

Wavefront OBJ

Reference

Menu:: *File ▶ Import/Export ▶ Wavefront (.obj)*

OBJ format is a popular plain text format, however, it has only basic geometry and material support.

Note

There is no support for armatures, lights, cameras, empty objects, parenting, or transformations. See [Compatibility](#) for more information.

Importing

Import geometry and curves to the OBJ format.

If there is a matching .MTL for the OBJ then its materials will be imported too.

General

Scale

Value by which to scale the imported objects in relation to the world's origin.

Clamp Bounding Box

OBJ-files often vary greatly in scale, this setting clamps the imported file to a fixed size.

Forward Axis, Up Axis

Since many applications use a different axis for ‘Up’, these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it’s common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Options

Split By Object

Import each OBJ “object name” group (o) as a separate object.

Split By Group

Import each OBJ “object name” group (g) as a separate object.

Vertex Groups

Import OBJ groups as vertex groups.

Validate Meshes

Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Path Separator

Character used to separate an object’s name into a hierarchical structure using [Collections](#).

Exporting

Export geometry and curves to the OBJ format.

General

Include: Selected Only

Only export the selected objects. Otherwise export all objects in the scene.

Scale

Global scale to use on export.

Forward Axis, Up Axis

Since many applications use a different axis for ‘Up’, there are axis conversion settings, Forward and Up axis – By mapping these to different axis you can convert rotations between applications default up and forward axis.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry Properties

UV Coordinates

Write out the active UV layers coordinates from Blender.

Normals

Write out Blender’s face and vertex normals (depending on the faces smooth setting).

Mostly this isn’t needed since most applications will calculate their own normals but to match Blender’s normal map textures you will need to write these too.

Colors

Write out the active vertex colors attribute layer, if present. Colors are exported in “xyzrgb” OBJ extension format.

Curves as NURBS

Write out NURBS curves as OBJ NURBS rather than converting to geometry.

Triangulated Mesh

Write out quads as two triangles. Some programs only have very basic OBJ support and only support triangles.

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

Properties

For properties that have different settings for the viewport/final render pick which is used for output. One example where this is important is the [Subdivision Surface Modifier](#).

Viewport::	Use viewport properties.
Render::	Use final render properties.

Grouping

Object Groups

Write out each Blender object as an OBJ object.

Note

Note that as far as Blender is concerned there is no difference between OBJ Groups and Objects, this option is only included for applications that treat them differently.

Material Groups

Generate an OBJ group for each part of a geometry using a different material.

Vertex Groups

Export the name of the vertex group of a face. It is approximated by choosing the vertex group with the most members among the vertices of a face.

Smooth Groups

Write Blender's sharp edges as smooth groups.

Smooth Group Bitflags

Generate Bitflags for smooth Groups.

Materials

Write out the MTL-file along with the OBJ. Most importers that support OBJ will also read the MTL-file.

PBR Extensions

Export MTL library using PBR extensions (roughness, metallic, sheen, clearcoat, anisotropy, transmission).

Path Mode

When referencing paths in exported files you may want some control as to the method used since absolute paths may only be correct on your own system. Relative paths, on the other hand, are more portable but mean that you have to keep your files grouped when moving about on your local file system. In some cases, the path doesn't matter since the target application will search a set of predefined paths anyway so you have the option to strip the path too.

Auto::	Uses relative paths for files which are in a subdirectory of the exported location, absolute for any directories outside that.
Absolute::	Uses full paths.
Relative::	Uses relative paths in every case (except when on a different drive on Windows).
Match::	Uses relative / absolute paths based on the paths used in Blender.
Strip Path::	Only write the filename and omit the path component.
Copy::	Copy the file on exporting and reference it with a relative path.

Animation

Exports a numbered OBJ for each frame from the start to the end frame. Please be aware that this can take quite a long time.

Frame Start, End

The first and last frame to export, used to determine the range of exported frames.

Compatibility

NURBS surfaces, text3D and metaballs are converted to meshes at export time.

Stanford PLY

Reference

Category:: Import-Export
Menu:: *File ▶ Import/Export ▶ Stanford (.ply)*

Use the operator to import ASCII or binary PLY-files, you can select multiple files at once. For exporting, you can choose to enable or disable the modifiers during the export and you can choose which data you want to export (UV textures, Color Attributes, ...).

Import

General

Scale

Value by which to scale the imported objects in relation to the world's origin.

Scene Unit

Apply current scene's unit (as defined by unit scale) to imported data.

Forward Axis, Up Axis

Since many applications use a different axis for 'Up', these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Options

Merge Vertices

Attempts to combine co-located vertices where possible.

Import Vertex Colors

The color space that the color data in the ply-file was saved in.

None::	Does not import vertex color data.
sRGB::	Vertex colors in the file are in sRGB Color Space
Linear::	Vertex colors in the file are in Linear Color Space

Export

General

Format: ASCII

Formats the file using the simple a ASCII format. This option might be helpful if the program that will later import the file does not support the binary file format.

Include: Selected Only

Only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered ‘selected’ when their instancer is selected.

Scale

Value by which to scale the exported objects in relation to the world’s origin.

Forward Axis, Up Axis

Since many applications use a different axis for ‘Up’, these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry

UV Coordinates

Write out the active UV layers coordinates from Blender.

Vertex Normals

Write out Blender's face and vertex normals (depending on the faces smooth setting).

Mostly this isn't needed since most applications will calculate their own normals but to match Blender's normal map textures you will need to write these too.

Vertex Colors

The color space that the color data in the ply-file was saved in.

None::	Does not import vertex color data.
sRGB::	Vertex colors in the file are in sRGB Color Space
Linear::	Vertex colors in the file are in Linear Color Space

Triangulated Mesh

All [N-gons](#) with four or more vertices will be triangulated. Meshes in the scene will not be affected. Behaves like [Triangulate Modifier](#) with the following settings:

- N-gon-method: "Beauty"
- Quad-method: "Shortest Diagonal"
- Min vertices: 4

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

STL

Reference

Category:: Import-Export
Menu:: *File ▶ Import/Export ▶ Stl (.stl)*

The STL-file format is useful if you intend to import/export the files for CAD software. It is also commonly used for loading into 3D printing software.

Importing

General

Scale

Value by which to scale the imported objects in relation to the world's origin.

Scene Unit

Apply current scene's unit (as defined by unit scale) to imported data.

Forward / Up Axis

Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, it is common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Options

Facet Normals

Use (import) facet normals (note that this will still give flat shading).

Validate Mesh

Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Exporting

General

Format: ASCII

Exports the stl-file in ASCII format rather than as a binary format

Batch

Export each object as a separate STL file.

Include: Selection Only

When checked, only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered 'selected' when their instancer is selected.

Scale

Value by which to scale the exported objects in relation to the world's origin.

Scene Unit

Apply current scene's unit (as defined by unit scale) to exported data.

Forward, Up

Since many applications use a different axis for 'Up', these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

Import/Export SVG as Grease Pencil

The Scalable Vector Graphics (SVG) format is use for interchanging vector based illustrations between applications and is supported by vector graphics editors such as Inkscape, and modern browsers among others.

Warning

The exporter only works in Object Mode.

Import

Reference

menu:: *File ▶ Import ▶ SVG as Grease Pencil*

Resolution

Resolution for generated strokes.

Scale

Generated strokes scale.

Export

Reference

menu:: *File ▶ Export ▶ Grease Pencil as SVG*

Scene Options

Object

Determine which objects include in the export.

Active::	Export only the active Grease Pencil object.
Selected::	Export all selected Grease Pencil objects.
Visible::	Export all visible Grease Pencil object in the scene.

Export Options

Sampling

Precision for the stroke sampling. Low values mean a more accurate result.

Fill

When enabled, Export the Grease Pencil strokes fill.

Uniform Width

When enabled, Export strokes with constant thickness.

Clip Camera

When enabled, and camera view is active export only the strokes clipped from camera view.

Export Grease Pencil as (PDF)

The Portable Document Format (PDF) is use for interchanging PDFs between applications, it support the export of Grease Pencil animation creating one page in the PDF document for each keyframe selected.

Warning

The exporter only works in Object Mode.

Scene Options

Object

Determine which objects will be included in the export.

Active::	Export only the active Grease Pencil object.
Selected::	Export all selected Grease Pencil objects.
Visible::	Export all visible Grease Pencil object in the scene.

Export Options

Frame

Determine which frames will be included in the export.

Active::	Export only the active keyframe.
Selected::	Export all selected keyframes as different PDF pages.
Scene::	Export all frames as different PDF pages.

Note

To enable multi-keyframe selection you must enable Multiframe Edition.
See [Multiframe Edition](#) for more information.

Sampling

Precision for the stroke sampling. Low values mean a more accurate result.

Fill

When enabled, Export the Grease Pencil strokes fill.

Uniform Width

When enabled, Export strokes with constant thickness.

Note

The export of the Grease Pencil strokes is always from camera view.

Add-ons

Important

This is work in progress.

Documentation might be outdated and on some pages images, videos, and links aren't added yet.

Add-ons Category Listings

- [3D View](#)
- [Animation](#)
- [Import-Export](#)
- [Node](#)
- [Rigging](#)
- [System](#)

3D View

These add-ons relate to drawing or manipulating the 3D Viewport.

- [VR Scene Inspection](#)

VR Scene Inspection

The VR Scene Inspection add-on exposes and extends the native virtual reality features of Blender in the user interface. The feature set is limited to scene inspection use cases. More advanced use cases may be enabled through further development inside of Blender.

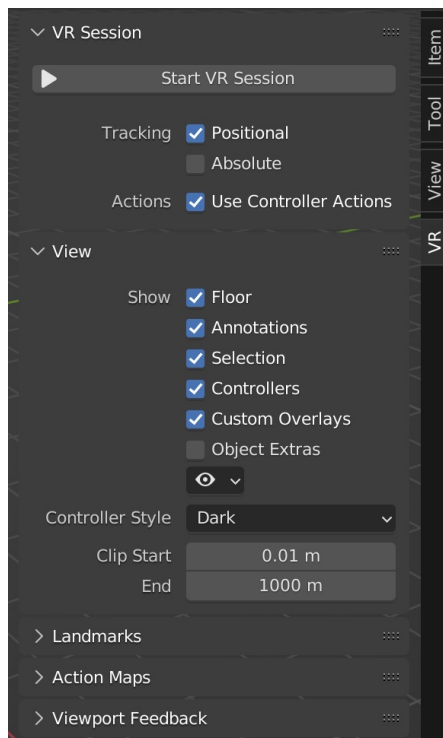
VR support in Blender is based on the OpenXR specification and requires some set up steps. These are explained in the [Head-Mounted Displays \(HMD\)](#) section.

Enabling Add-on

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “VR Scene Inspection” and check the *Enable Extension* checkbox.

Interface

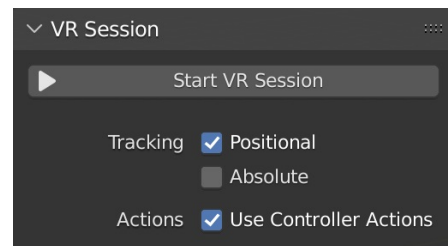
Located in the *3D Viewport* ▶ *Sidebar* ▶ *VR tab*.



VR Session

Start VR Session

Try to set up a connection to the OpenXR platform to share the viewport with an [HMD](#).



Tracking

Positional

Only track rotational changes of the head, do not allow the HMD to affect the location of the viewer in virtual space.

Absolute

Skip eye offsets that are normally added for placing the viewer exactly at landmarks. This allows the tracking origin to be defined independently of the HMD position.

Use Controller Actions

Enable default controller actions for viewport navigation, controller tracking, and haptics.

View

Show

Floor

Set visibility of the ground plane in the VR view.

Annotations

Set visibility of annotation strokes in the VR view.

Selection

Set visibility of selection outlines in the VR view.

Controllers

Set visibility of VR motion controllers. Requires enabling the [Use Controller Actions](#) option.

Custom Overlays

Set visibility of custom operator drawing (e.g. default teleport beam).

Object Extras

Set visibility of object extras, including empties, lights, and cameras.

Object Type Visibility

Set visibility of objects by type.

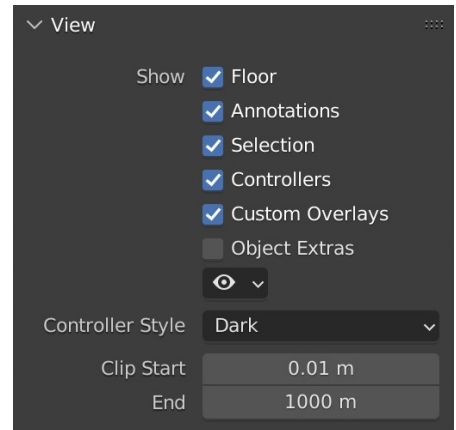
Controller Style

Preferred visualization of VR motion controllers.

Clip Start/End

Clipping values of the VR view, [as in the 3D Viewport](#).

Landmarks



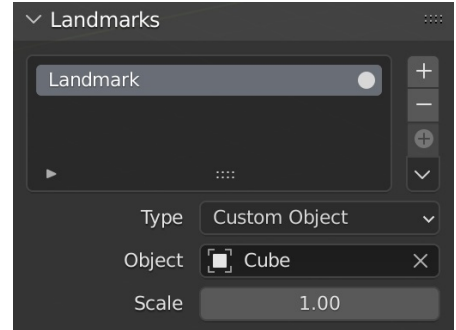
Landmarks are used to store reusable base poses (position and rotation) for the viewer in the virtual space. In addition, a base viewer reference scale can be set for landmarks of types Custom Object and Custom Pose.

Landmark

A [list view](#).

Selected Landmark

Defines which landmark's settings are shown below the list. Changing the selected landmark does not have an influence on the VR view.



Activate ○

Activates a landmark, making it change the base pose of the VR view.

Add +

Create a landmark.

Remove –

Delete the selected landmark.

Add from Session ⊕

Create a landmark from the viewer pose of the running VR session.

Landmark Controls v

Add Camera and VR Landmark from Session

Create a new camera and landmark from the viewer pose of the running VR session.

Add Landmark from Camera

Add a new landmark from the active camera object.

Update Custom Landmark

Update the selected landmark from the current VR viewer pose.

Cursor to Landmark

Move the 3D Cursor to the selected landmark.

Scene Camera to Landmark

Position the scene camera at the selected landmark.

Camera from Landmark

Create a new camera from the selected landmark.

Type

Scene Camera

Follow the scene's [active camera](#) to define the base pose of the viewer.

Custom Object

Set an arbitrary object to define the base pose of the viewer.

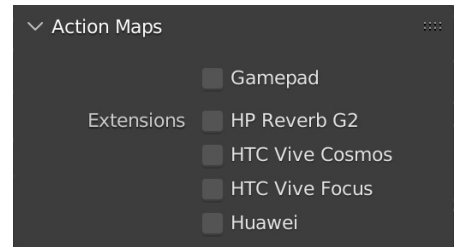
Custom Pose

Manually define a position and rotation to use as the base pose of the viewer.

Action Maps

Gamepad

Use input from a gamepad (Microsoft Xbox Controller) instead of motion controllers for VR actions such as viewport navigation.



Extensions

Enable additional controller bindings to ensure correct input-to-action mappings. Note that a given extension may not be supported by all [VR platforms](#).

HP Reverb G2

Enable bindings for the HP Reverb G2 controllers.

HTC Vive Cosmos

Enable bindings for the HTC Vive Cosmos controllers.

HTC Vive Focus

Enable bindings for the HTC Vive Focus 3 controllers.

Huawei

Enable bindings for the Huawei controllers.

Viewport Feedback

Show VR Camera

Draw an indicator of the current VR viewer pose (location and rotation in the virtual space) in the current 3D Viewport.

Show VR Controllers

Draw indicators of tracked VR motion controllers in the current 3D viewport.

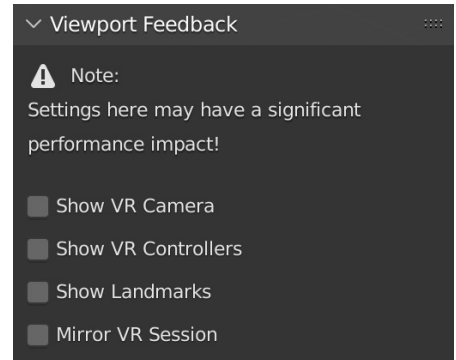
Requires enabling the [Use Controller Actions](#) option.

Show Landmarks

Draw [landmark](#) indicators in the current 3D Viewport.

Mirror VR Session

Make the current 3D Viewport follow the perspective of the VR view.



Reference

Category::	3D View
Description::	View the viewport with virtual reality glasses (head-mounted displays).
Location::	<i>3D Viewport ▶ Sidebar ▶ VR tab</i>
File::	viewport_vr_preview folder
Author::	Julian Eisel, Sebastian Koenig, Peter Kim
Maintainer::	Julian Eisel, Peter Kim
License::	GPL
Support Level::	Official
Note::	This add-on is bundled with Blender.

Animation

These add-ons relate to helper tools for the animation process and animation.

- [Copy Global Transform](#)

Copy Global Transform

Copy and paste object and bone transforms with ease.

When copying, the global ([World Space](#)) transform is placed on the clipboard. This can then be pasted onto any object or bone, at the current frame or at another one.

Since the transform is placed on the clipboard as text, you can even copy-paste it into an instant messenger and send it to someone else.

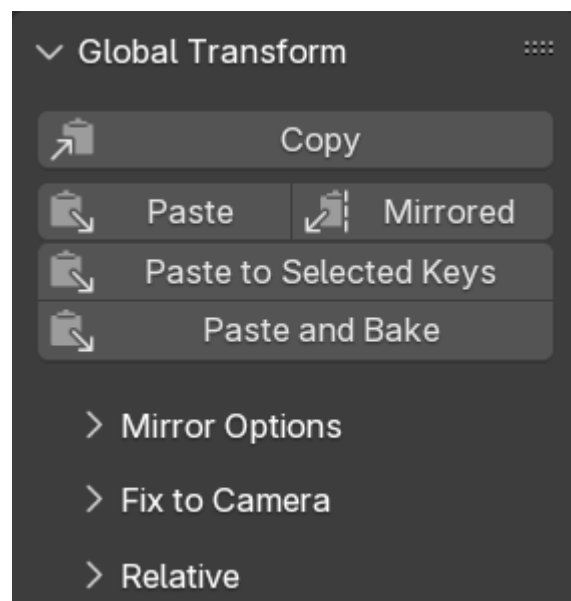
Enabling Add-on

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “Copy Global Transform” and check the *Enable Extension* checkbox.

Interface

Located in *3D Viewport* ▶ *N-panel* ▶ *Animation* tab.

The figure on the right shows the main functionality of the Copy Global Transform panel. The collapsed panels are described each in their own section below.



Description

Copy

Inspects the active Object (in Object mode) or Bone (in Pose mode) and places its current global transform onto the clipboard as a matrix.

Paste

Takes the copied global transform and applies it to the active Object or Bone. This is done by **adjusting its location, rotation, and scale properties**.

Mirrored

Same as 'Paste' above, but then mirrored relative to some other object or bone. This can be useful, for example, to copy the foot position of one foot to the other. See [Mirror Options](#) below.

Paste to Selected Keys

Paste as described above and additionally use auto-keying to update one or more frames. The key selection is used to tell Blender *which frames* this should happen on; it does not influence which parts of the transform are keyed. *What* is keyed is determined by the active keying set.

Paste and Bake

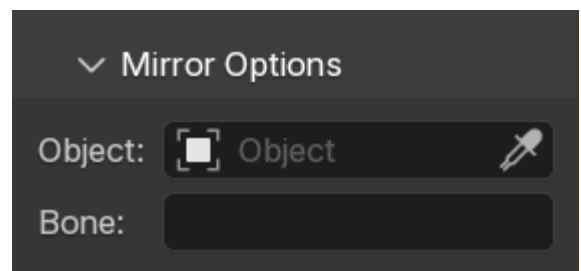
Almost the same as *Paste to Selected Keys*. Instead of only pasting on the selected keys, *Paste and Bake* will paste & auto-key on every frame between the first and last selected keys.

Mirror Options

The copied transform can be mirrored relative to an object or a [Bone](#). This requires choosing that object or bone first.

Armature + Bone

Choosing an [Armature](#) object as mirror object will show the bone



selector. You can use that to pick the bone to use as mirror. This will always use the named bone on that specific armature object.

Bone Only

When you choose *no mirror object* at all, you can still choose a *bone name*. This is used for mirroring against a bone in the *active armature*. This can be useful to mirror bone transforms relative to the ‘chest’ bone of the active character.

Object Only

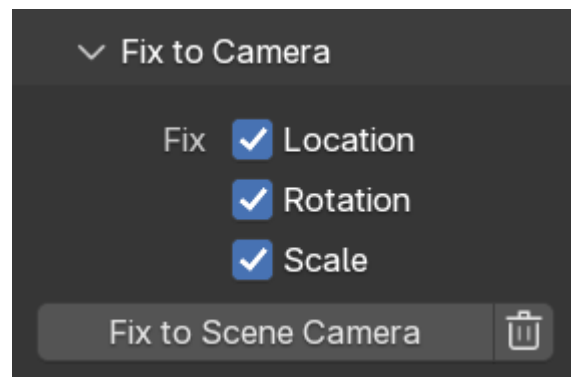
This will just mirror relative to the chosen object.

After pasting with ‘Paste Mirrored’, the mirror axes can be chosen in the [redo panel](#).

Fix to Camera

Also known as “bake to camera”, this operator will ensure selected objects/bones remain static (relative to the camera) on unkeyed frames.

This is done by generating new keys. These keys will be of [type ‘Generated’](#) so that it remains clear which keys were manually created, and which were generated. This way the tool can be re-run to re-generate the keys.



1. Ensure your animation is keyed using constant interpolation. If this is not the case yet, bake your animation (at least the transform channels). This tool does not work with the [“Stepped” F-Curve modifier](#)
2. Choose which of the Location/Rotation/Scale channels you want to fix to the camera. When unsure, make sure they are all checked.
3. Press the “Fix to Camera” button.

To undo the effect of the “Fix to Camera” operator, click on the trash bin button. That will remove all the generated keys in either the scene range or the frame range.

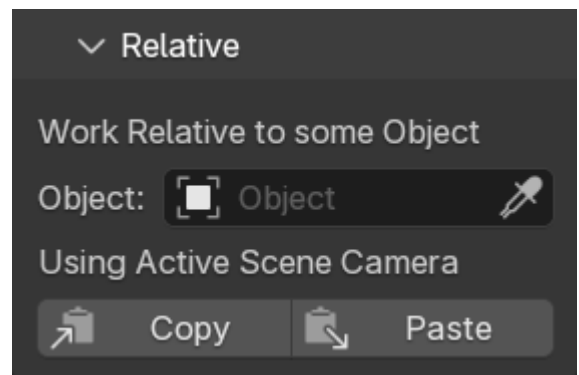
The tool operates on the scene frame range, or on the preview range if that is active. Keys outside that range are ignored, both when fixing to the camera and when removing generated keys.

Warning

This tool assumes that *all* keys with type ‘Generated’ are equal. It will overwrite them (or remove them, depending on which button you press).

Relative Copy-Paste

The “Relative” panel has copy/paste buttons that work relative to a chosen object. When copying, the world-space transform is determined, and then adjusted to become relative to the world-space transform of the chosen object. When pasting, this happens in reverse.



If no object is chosen, the copy/paste will happen relative to the active scene camera. What is the active scene camera is determined for every action, so when you paste it can be different from when you copied. This can help to keep an object visually in the same place when switching cameras, or when switching between scenes.

Limitations

Pasting a transform adjusts the Object/Bone’s location, rotation, and scale. This means that when copying a skewed transform, this skew is lost.

If there are constraints on the Object/Bone, the resulting visual transformation may not be the same as the pasted one. To give a concrete example: if you have a constraint that adds a rotation, it will always add that rotation on top of the pasted transform.

See also

[Pose Library](#) for a way to manage and share entire poses.

Reference

Category::	Animation
Description::	Simple add-on for copying world-space transforms.
Location::	<i>3D Viewport ▶ N-panel ▶ Animation tab</i>
File::	copy_global_transform.py
Author::	Sybren A. Stüvel
Maintainer::	Sybren A. Stüvel
License::	GPL 2+
Support Level::	Official
Note::	This add-on is bundled with Blender.

Import-Export

- [BioVision Motion Capture \(BVH\)](#)
- [FBX](#)
- [Scalable Vector Graphics \(SVG\)](#)
- [UV Layout](#)
- [glTF 2.0](#)

BioVision Motion Capture (BVH)

Reference

Category:: Import-Export
Menu:: *File ▶ Import/Export ▶ Motion Capture (.bvh)*

Imports or exports bvh-files or files with BioVision Hierarchical data or data of a skeleton (rig) including its animation. Useful for importing data from motion capture devices.

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “BioVision Motion Capture (BVH) format” and check the *Enable Extension* checkbox.

Properties

Import

Target

The motion capture data type.

Armature:: The bvh-file contains an animated rigged skeleton such as a walking motion capture.

Object:: The bvh-file contains a static (not animated) mesh object such as a character model.

Transform

Scale

Factor to increase the physical size of the BVH.

Rotation

Rotation order of the BVH.

Forward / Up

Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Animation

Start Frame

The start frame, in Blender, to start playback of the BVH animation.

Scale FPS

Scales the frame rate from the BVH file to the scene frame rate set in Blender, otherwise each BVH frame maps directly to a frame in Blender.

Loop

Cycles the animation playback.

Update Scene FPS

Set the scene's frame rate to match the frame rate of the BVH file.

Update Scene Duration

Extend the scene's duration to match the BVH's duration.

Export

Transform

Scale

Factor to increase the physical size of the BVH.

Rotation

Rotation order of the BVH.

Root Translation Only

Only write the translation animation channels for the root bone.

Animation

Start / End

Sets the range of animation to export to the BVH file.

FBX

Reference

Category:: Import-Export
Menu:: *File ▶ Import/Export ▶ FBX (.fbx)*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “FBX” and check the *Enable Extension* checkbox.

Usage

This format is mainly use for interchanging character animations between applications and is supported by applications such as Cinema4D, Maya, Autodesk 3ds Max, Wings3D and engines such as Unity3D, Unreal Engine 3/UDK and Unreal Engine 4.

The exporter can bake mesh modifiers and animation into the FBX so the final result looks the same as in Blender.

Note

- Bones would need to get a correction to their orientation (FBX bones seems to be -X aligned, Blender’s are Y aligned), this does not affect skinning or animation, but imported bones in other applications will look wrong.
- Animations (FBX AnimStacks, Blender actions) **are not linked** to their object, because there is no real way to know which stack to use as ‘active’ action for a given object, mesh or bone. This may be

enhanced to be smarter in the future, but it's not really considered urgent, so for now you'll have to link actions to objects manually.

- Armature instances **are not supported**.

Note

- Bones' orientation importing is complex, you may have to play a bit with related settings until you get the expected results.
- Animation support is minimal currently, we read all curves as if they were 'baked' ones (i.e. a set of close keyframes with linear interpolation).
- Imported actions are linked to their related object, bone or shape key, on a 'first one wins' basis. If you export a set of them for a single object, you'll have to reassign them yourself.

Note

Saving Just Animations

The FBX file format supports files that only contain takes. It is up to you to keep track of which animation belongs to which model. The animation that will be exported is the currently selected action within the Action editor. To reduce the file size, turn off the exporting of any parts you do not want and disable *All Actions*. For armature animations typically you just leave the armature enabled which is necessary for that type of animation. Reducing what is output makes the export and future import much faster. Normally each action will have its own name but the current or only take can be forced to be named "Default Take". Typically, this option can remain off.

Note

Blender now only supports complex node-based shading. FBX having a fixed pipeline-like support of materials, this add-on converts between them.

Properties

Import

Include

Import Normals
TODO.

Import Subdivision Surface
Todo.

Import User Properties
TODO.

Import Enums as Strings
TODO.

Image Search
TODO.

Transform

Scale
Todo.

Decal Offset
TODO.

Manual Orientation
TODO.

Forward / Up Axis

Since many applications use a different axis for 'Up', these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Apply Transform

TODO.

Use Pre/Post Rotation

TODO.

Animation

TODO.

Animation Offset

TODO.

Armature

Ignore Leaf Bones

TODO.

Force Connect Children

TODO.

Automatic Bone Orientation

TODO.

Primary/Secondary Bone Axis

TODO.

Export

Path Mode

When referencing paths in exported files you may want some control as to the method used since absolute paths may only be correct on your own system. Relative paths, on the other hand, are more portable but mean that you have to keep your files grouped when moving about on your local file system. In some cases, the path doesn't matter since the target application will search a set of predefined paths anyway so you have the option to strip the path too.

Auto::	Uses relative paths for files which are in a subdirectory of the exported location, absolute for any directories outside that.
Absolute::	Uses full paths.
Relative::	Uses relative paths in every case (except when on a different drive on Windows).
Match::	Uses relative / absolute paths based on the paths used in Blender.
Strip Path::	Only write the filename and omit the path component.
Copy::	Copy the file on exporting and reference it with a relative path.

Embed Textures
TODO.

Batch Mode

When enabled, export each group or scene to a file.

Group/Scene

Choose whether to batch export groups or scenes to files. Note, when Group/Scene is enabled, you cannot use the animation option *Current Action* since it uses scene data and groups are not attached to any scenes. Also note, when Group/Scene is enabled you must include the armature objects in the group for animated actions to work.

Batch Own Directory

When enabled, each file is exported into its own directory, this is useful when using the *Copy Images* option. So each directory contains one model with all the images it uses. Note, this requires a full Python installation. If you do not have a full Python installation, this button will not be shown.

Include

Selected Objects

Only export the selected objects. Otherwise export all objects in the scene. Note, this does not apply when batch exporting.

Active Collection

Todo.

Object Types

Enable/Disable exporting of respective object types.

Custom Properties

TODO.

Transform

Scale

Scale the exported data by this value. 10 is the default because this fits best with the scale most applications import FBX to.

Apply Scaling

TODO.

Forward / Up

Since many applications use a different axis for 'Up', these are axis conversions for Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Apply Unit

TODO.

Apply Transform

TODO.

Geometry

Smoothing

TODO.

Export Subdivision Surface

Todo.

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

Loose Edges

TODO.

Tangent Space

TODO.

Armatures

Primary/Secondary Bone Axis

TODO.

Armature FBXNode Type

TODO.

Only Deform Bones

TODO.

Add Leaf Bones
TODO.

Bake Animation

TODO.

Key All Bones
TODO.

NLA Strips
TODO.

All Actions

Export all actions compatible with the selected armatures start/end times which are derived from the keyframe range of each action. When disabled only the currently assigned action is exported.

Force Start/End Keying
TODO.

Sampling Rate
TODO.

Simplify
TODO.

Compatibility

Import

Note that the importer is a new addition and lacks many features the exporter supports.

- binary FBX files only.
- Version 7.1 or newer.

Missing

- Mesh: shape keys.

Export

NURBS surfaces, text3D and metaballs are converted to meshes at export time.

Missing

Some of the following features are missing because they are not supported by the FBX format, others may be added later.

- Object instancing – exported objects do not share data, instanced objects will each be written with their own data.
- Material textures
- Vertex shape keys – FBX supports them but this exporter does not write them yet.
- Animated fluid simulation – FBX does not support this kind of animation. You can however use the OBJ exporter to write a sequence of files.
- Constraints – The result of using constraints is exported as a keyframe animation however the constraints themselves are not saved in the FBX.
- Instanced objects – At the moment instanced objects are only written in static scenes (when animation is disabled).

Scalable Vector Graphics (SVG)

Reference

Category:: Import-Export

Menu:: *File ▶ Import ▶ Scalable Vector Graphics (.svg)*

Note

Currently the script allows only importing and is limited to path geometry only.

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “Scalable Vector Graphics (SVG)” and check the *Enable Extension* checkbox.

Properties

This add-on does not have any properties.

Usage

Todo.

UV Layout

Reference

Category:: Import-Export
Menu:: *UV Editor ▸ UV ▸ Export UV Layout*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “UV Layout” and check the *Enable Extension* checkbox.

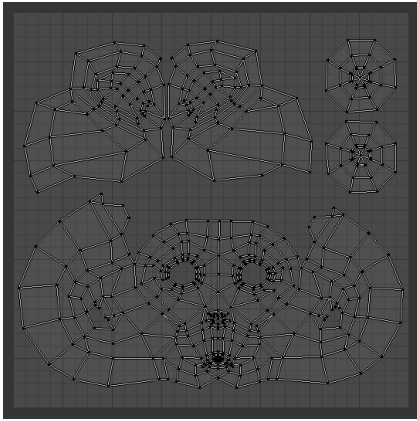
Usage

Using your favorite image painting program, you could use an exported UV layout to create a texture. Then save your changes, and back in Blender, use the *Image ▸ Open* to load it as your UV image for the mesh in Edit Mode for the desired (and active) UV map.

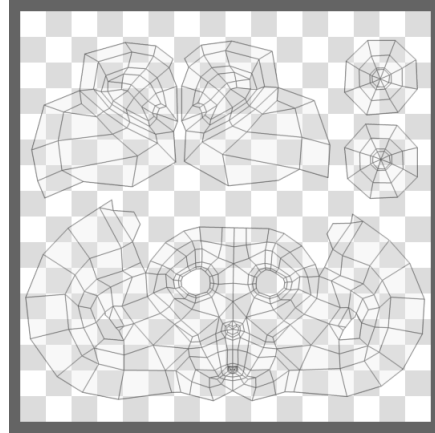
As a way of communicating to an artist who is painting your UV Texture for you, Blender has a tool called *UV Layout* (*UV Editor ▸ UV ▸ Export UV Layout*) that saves an image as a Targa (.tga), EPS, or SVG format for the object you have selected.

The image will be lines defining the UV edges that are within the image area of the UV mapping area. Edges outside the boundary, even if selected, will not be shown in the saved graphic. The artist will use this as a transparent layer in their paint program as a guide when painting your texture. The example below shows Blender in the background, and the Gimp working on the texture, using the saved layout as a guide. Note that

targa format supports the Alpha channel, so you can paint transparent areas of the mesh. For using images as textures, see the page on [Image Textures](#).

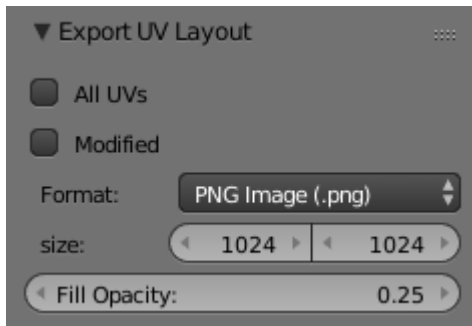


A UV layout in the UV Editor.



A UV layout in a paint program.

Properties



Export options.

All UVs

if disabled, then only the UV faces selected will be outlined.

Modified

Export UVs from the modified mesh.

Format

Select the type of image file to save (.png, .eps, .svg).

Size

Select the size of the image in pixels.

Fill Opacity

Set the opacity of the fill.

glTF 2.0

Reference

Category:: Import-Export
Menu:: *File ▶ Import/Export ▶ glTF 2.0 (.glb, .gltf)*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “glTF 2.0” and check the *Enable Extension* checkbox.

Usage

glTF™ (GL Transmission Format) is used for transmission and loading of 3D models in web and native applications. glTF reduces the size of 3D models and the runtime processing needed to unpack and render those models. This format is commonly used on the web, and has support in various 3D engines such as Unity3D, Unreal Engine 4, and Godot.

This importer/exporter supports the following glTF 2.0 features:

- Meshes
- Materials (Principled BSDF) and Shadeless (Unlit)
- Textures
- Cameras
- Punctual lights (point, spot, and directional)
- Extensions (listed below)
- Extras (custom properties)
- Animation (keyframe, shape key, and skinning)

Meshes

glTF's internal structure mimics the memory buffers commonly used by graphics chips when rendering in real-time, such that assets can be delivered to desktop, web, or mobile clients and be promptly displayed with minimal processing. As a result, quads and n-gons are automatically converted to triangles when exporting to glTF. Discontinuous UVs and flat-shaded edges may result in moderately higher vertex counts in glTF compared to Blender, as such vertices are separated for export. Likewise, curves and other non-mesh data are not preserved, and must be converted to meshes prior to export.

GPU Instances

When the option is enable in Exporter, instances are exported using the EXT_mesh_gpu_instancing extension. There are some limitations, at export:

- Instances must be meshes, and don't have any children themselves
- Instances must all be children of the same object.
- This extension doesn't manage material variation. That means that the generated file may include all instances with same materials.
- Instances detected are objects sharing the same mesh data.

At import, instances are created by creating objects sharing the same mesh data.

Materials

The core material system in glTF supports a metal/rough PBR workflow with the following channels of information:

- Base Color
- Metallic
- Roughness
- Baked Ambient Occlusion

- Normal Map (tangent space, +Y up)
- Emissive

Some additional material properties or types of materials can be expressed using glTF extensions. The complete list can be found in [Extensions](#) part of this documentation.



An example of the various image maps available in the glTF 2.0 core format. This is the [water bottle sample model](#) shown alongside slices of its various image maps.

Imported Materials

The glTF material system is different from Blender's own materials. When a glTF file is imported, the add-on will construct a set of Blender nodes to replicate each glTF material as closely as possible.

The importer supports Metal/Rough PBR (core glTF), Spec/Gloss PBR (KHR_materials_pbrSpecularGlossiness) and some extension materials. The complete list can be found in [Extensions](#) part of this documentation.

Tip

Examining the result of the material import process is a good way to see examples of the types of material nodes and settings that can be exported to glTF.

Exported Materials

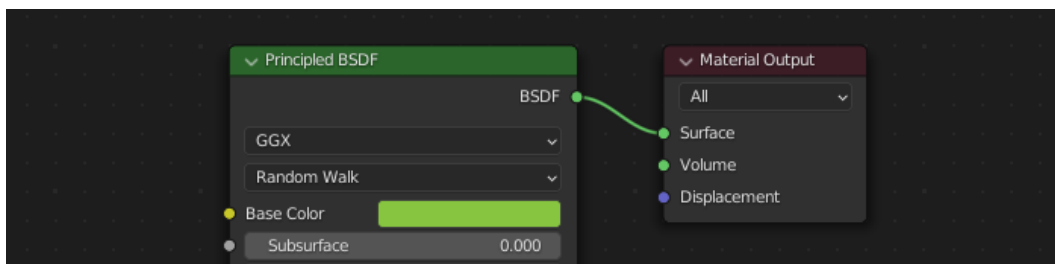
The exporter supports Metal/Rough PBR (core glTF) and Shadeless (KHR_materials_unlit) materials. It will construct a glTF material based on the nodes it recognizes in the Blender material. The material export process handles the settings described below.

Note

When image textures are used by materials, glTF requires that images be in PNG or JPEG format. The add-on will automatically convert images from other formats, increasing export time.

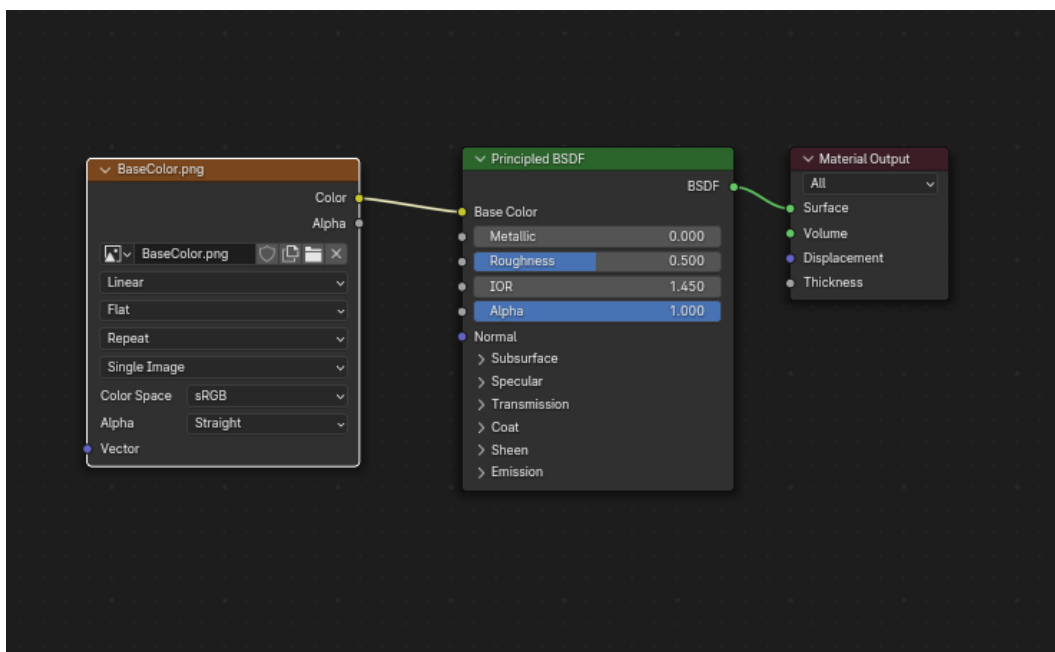
Base Color

The glTF base color is determined by looking for a Base Color input on a Principled BSDF node. If the input is unconnected, the input's default color (the color field next to the unconnected socket) is used as the Base Color for the glTF material.



A solid base color can be specified directly on the node.

If an Image Texture node is found to be connected to the Base Color input, that image will be used as the glTF base color.



An image is used as the glTF base color.

Metallic and Roughness

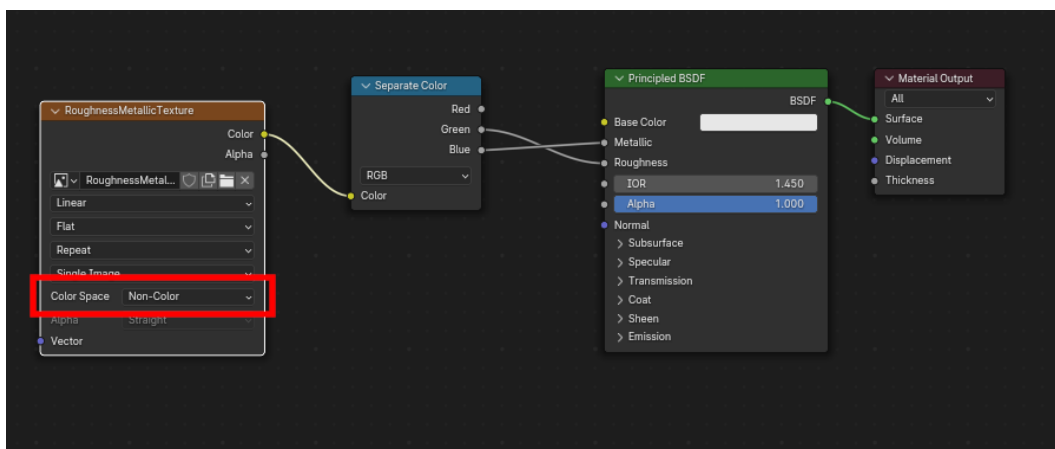
These values are read from the Principled BSDF node. If both of these inputs are unconnected, the node will display sliders to control their respective values between 0.0 and 1.0, and these values will be copied into the glTF.

When using an image, glTF expects the metallic values to be encoded in the blue (B) channel, and roughness to be encoded in the green (G) channel of the same image. If images are connected to the Blender node in a manner that does not follow this convention, the add-on may attempt to adapt the image to the correct form during exporting (with an increased export time).

In the Blender node tree, it is recommended to use a Separate RGB node to separate the channels from an Image Texture node, and connect the green (G) channel to Roughness, and blue (B) to Metallic. The glTF exporter will

recognize this arrangement as matching the glTF standard, and that will allow it to simply copy the image texture into the glTF file during export.

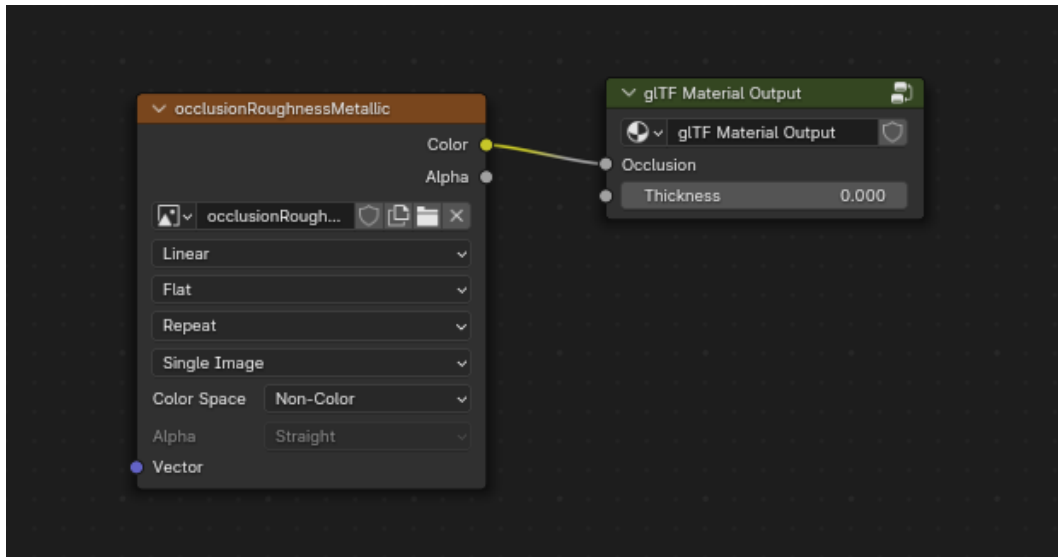
The Image Texture node for this should have its *Color Space* set to Non-Color.



A metallic/roughness image connected in a manner consistent with the glTF standard, allowing it to be used verbatim inside an exported glTF file.

Baked Ambient Occlusion

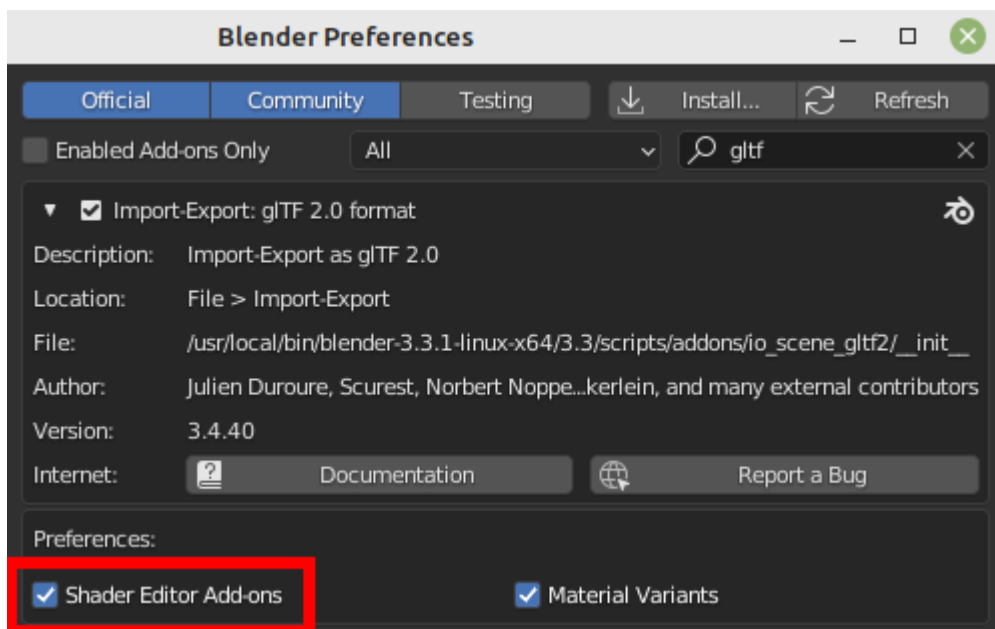
glTF is capable of storing a baked ambient occlusion map. Currently there is no arrangement of nodes that causes Blender to use such a map in exactly the same way as intended in glTF. However, if the exporter finds a custom node group by the name of glTF Material Output, and finds an input named Occlusion on that node group, it will look for an Image Texture attached there to use as the occlusion map in glTF. The effect need not be shown in Blender, as Blender has other ways of showing ambient occlusion, but this method will allow the exporter to write an occlusion image to the glTF. This can be useful to real-time glTF viewers, particularly on platforms where there may not be spare power for computing such things at render time.



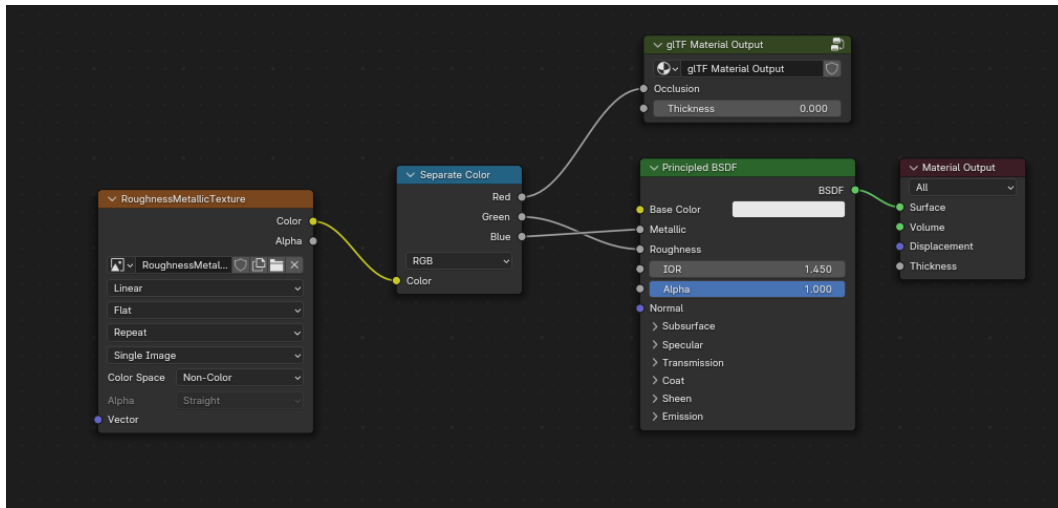
A pre-baked ambient occlusion map, connected to a node that doesn't render but will export to glTF.

Tip

If you enable Shader Editor Add-ons in preferences, you will be able to add this custom node group from Menu: Add > Output > glTF Material Output



glTF stores occlusion in the red (R) channel, allowing it to optionally share the same image with the roughness and metallic channels.



This combination of nodes mimics the way glTF packs occlusion, roughness, and metallic values into a single image.

Tip

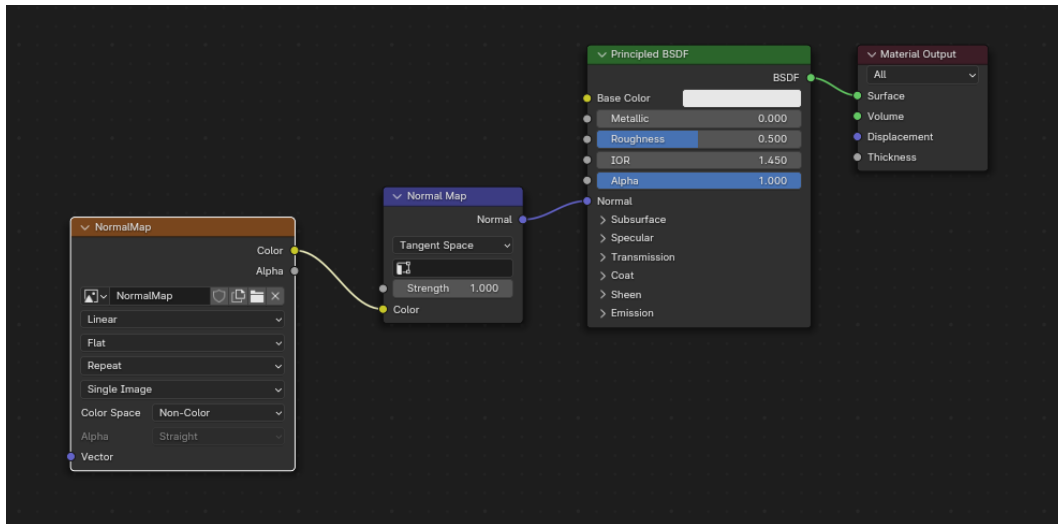
The Cycles render engine has a Bake panel that can be used to bake ambient occlusion maps. The resulting image can be saved and connected directly to the glTF Material Output node.

Normal Map

To use a normal map in glTF, connect an Image Texture node's color output to a Normal Map node's color input, and then connect the Normal Map normal output to the Principled BSDF node's normal input. The Image Texture node for this should have its *Color Space* property set to Non-Color.

The Normal Map node must remain on its default property of Tangent Space as this is the only type of normal map currently supported by glTF. The strength of the normal map can be adjusted on this node. The exporter

is not exporting these nodes directly, but will use them to locate the correct image and will copy the strength setting into the glTF.



A normal map image connected such that the exporter will find it and copy it to the glTF file.

Tip

The Cycles render engine has a Bake panel that can be used to bake tangent-space normal maps from almost any other arrangement of normal vector nodes. Switch the Bake type to Normal. Keep the default space settings (space: Tangent, R: +X, G: +Y, B: +Z) when using this bake panel for glTF. The resulting baked image can be saved and plugged into to a new material using the Normal Map node as described above, allowing it to export correctly.

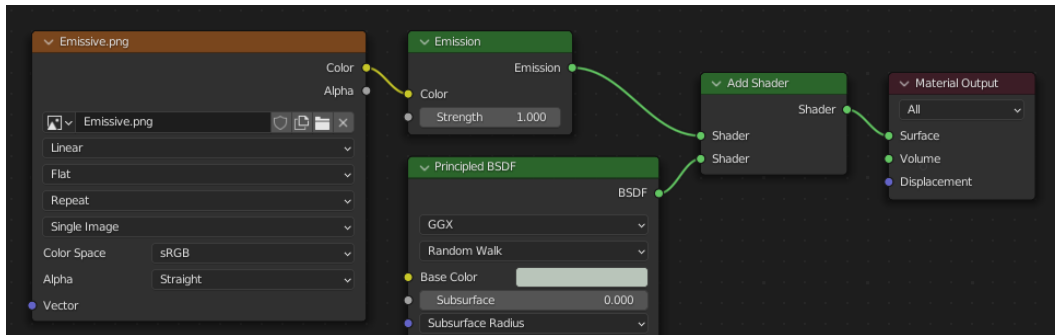
See: [Cycles Render Baking](#)

Emissive

An Image Texture node can be connected to the Emission input on the Principled BSDF node to include an emissive map with the glTF material. Alternatively, the Image Texture node can be connected to an Emission

shader node, and optionally combined with properties from a Principled BSDF node by way of an Add Shader node.

If the emissive map is alone in the material, it is best to set the Base Color default to black, and the Roughness default to 1.0. This minimizes the influence of the other channels if they are not needed.



This arrangement is supported for backwards compatibility. It is simpler to use the Principled BSDF node directly.

If any component of emissiveFactor is > 1.0 ,
KHR_materials_emissive_strength extension will be used.

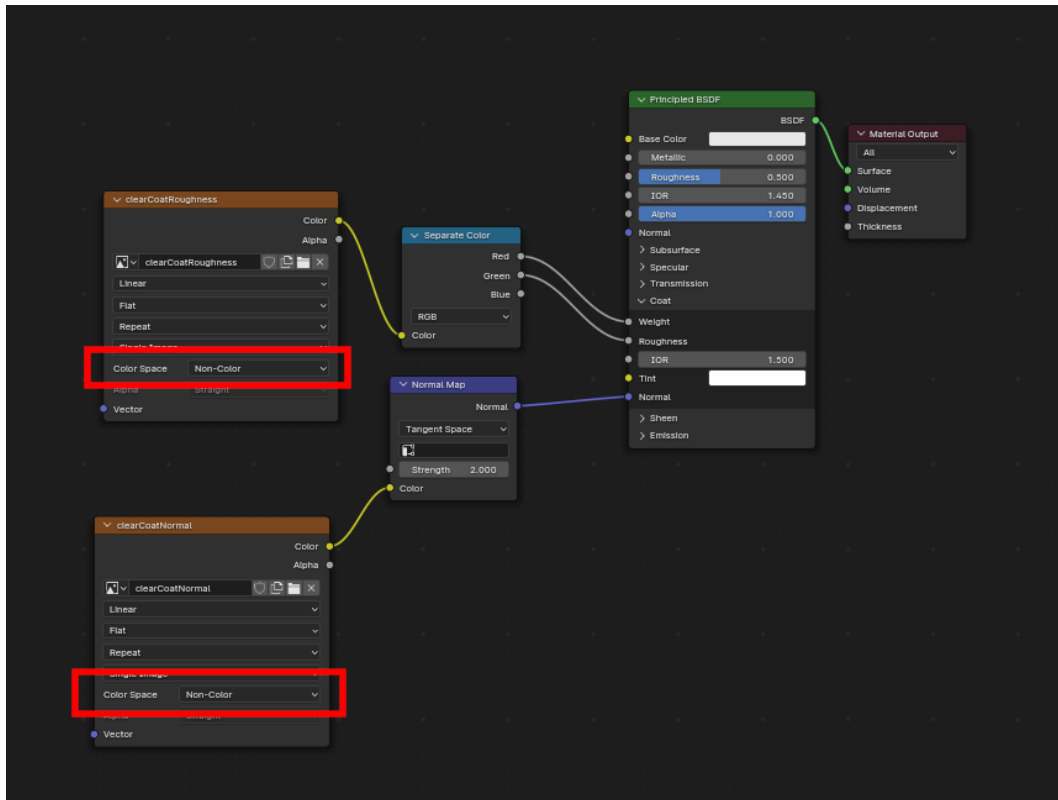
Clearcoat

When the *Clearcoat* input on the Principled BSDF node has a nonzero default value or Image Texture node connected, the KHR_materials_clearcoat glTF extension will be included in the export. This extension will also include a value or Image Texture from the *Clearcoat Roughness* input if available.

If Image Textures are used, glTF requires that the clearcoat values be written to the red (R) channel, and *Clearcoat Roughness* to the green (G) channel. If monochrome images are connected, the exporter will remap them to these color channels.

The *Clearcoat Normal* input accepts the same kinds of inputs as the base Normal input, specifically a tangent-space normal map with +Y up, and a user-defined strength. This input can reuse the same normal map that the

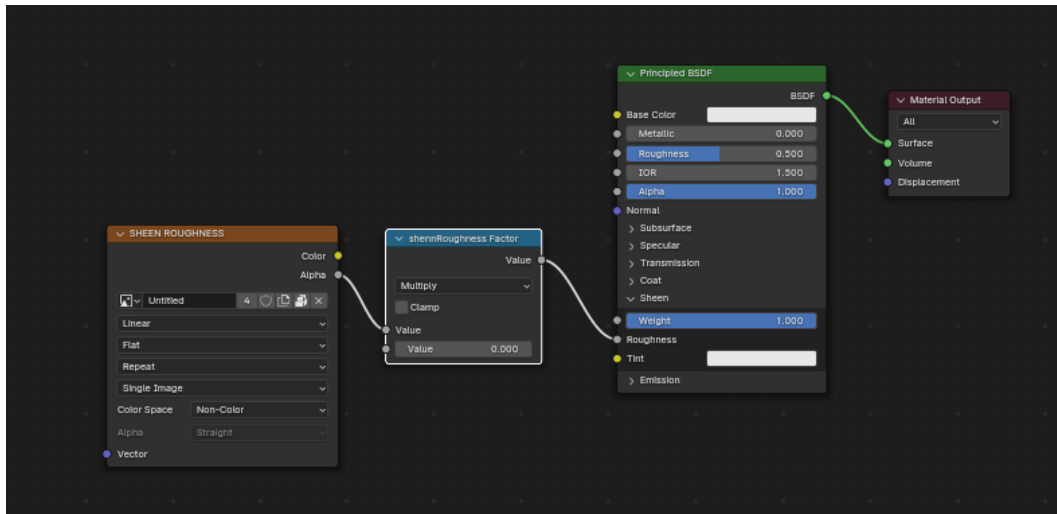
All Image Texture nodes used for clearcoat shading should have their *Color Space* set to Non-Color.



An example of a complex clearcoat application that will export correctly to glTF. A much simpler, smooth coating can be applied from just the Principled BSDF node alone.

Sheen

If a Sheen Roughness Texture is used, glTF requires the values be written to the alpha (A) channel.



Tip

Sheen BSDF node is only available on Cycles render engine. You may have to temporary switch to Cycles to add this node, and get back to EEVEE.

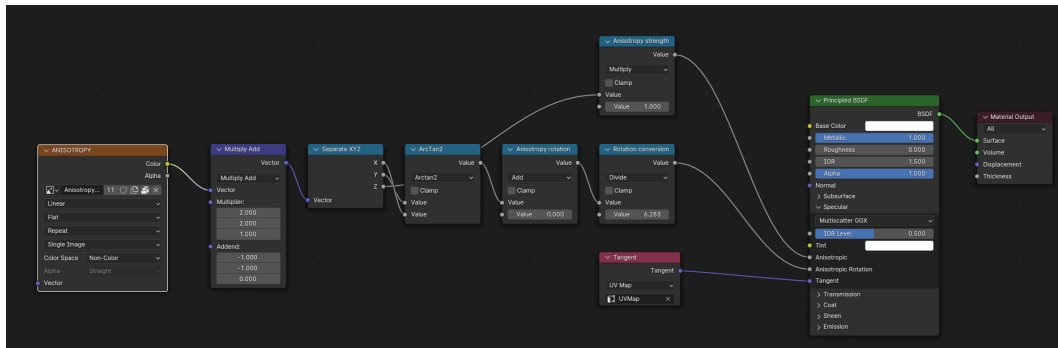
Specular

When the *Specular IOR Level* or *Specular Tint* input of Principled BSDF node have a non default value or Image Texture node connected, the KHR_materials_specular glTF extension will be included in the export.

Anisotropy

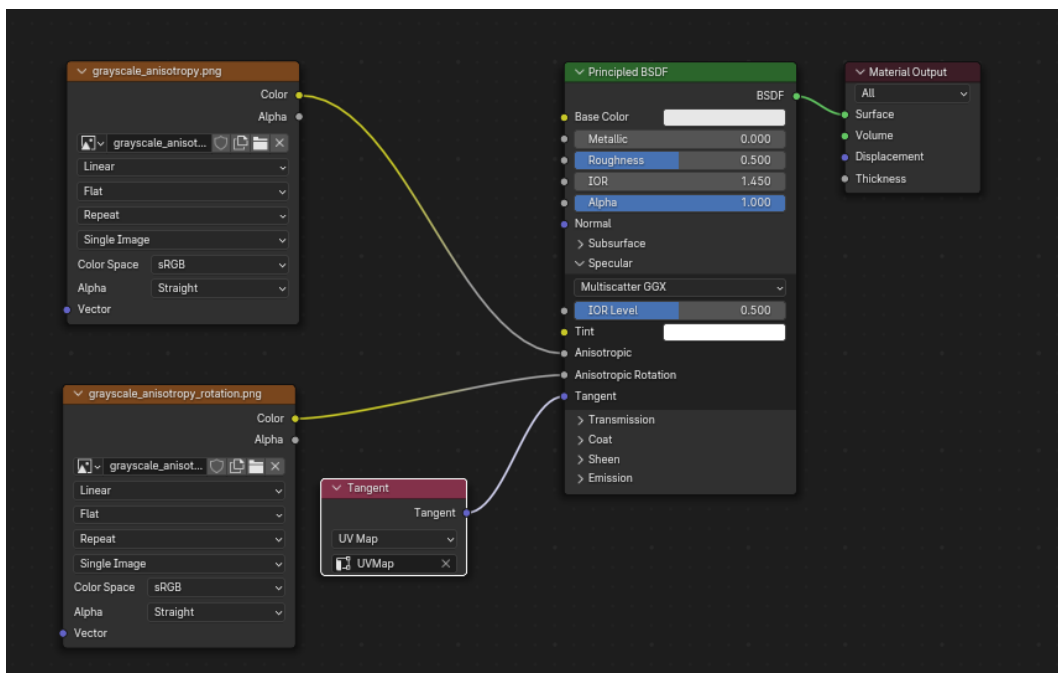
Anisotropic textures and data need to be converted at export, and at import.

At import, some nodes are created to manage this conversion



At export, this exact same nodes are detected, and used to export data.

At export, you can also plug some grayscale textures for *Anisotropic* and *Anisotropic Rotation* sockets. Then, exporter will convert these texture into a glTF compatible texture.



Note that the *tangent* socket must be linked to a *tangent* node, with UVMap. The chosen UVMap must be the UVMap of the Normal Map.

Transmission

When the Transmission input on the Principled BSDF node has a nonzero default value or Image Texture node connected, the

KHR_materials_transmission glTF extension will be included in the export. When a texture is used, glTF stores the values in the red (R) channel. The *Color Space* should be set to Non-Color.

Transmission is different from alpha blending, because transmission allows full-strength specular reflections. In glTF, alpha blending is intended to represent physical materials that are partially missing from the specified geometry, such as medical gauze wrap. Transmission is intended to represent physical materials that are solid but allow non-specularly-reflected light to transmit through the material, like glass.

Tip

The material's base roughness can be used to blur the transmission, like frosted glass.

Tip

Typically the alpha blend mode of a transmissive material should remain “Opaque”, the default setting, unless the material only partially covers the specified geometry.

Note

In real-time engines where transmission is supported, various technical limitations in the engine may determine which parts of the scene are visible through the transmissive surface. In particular, transmissive materials may not be visible behind other transmissive materials. These limitations affect physically-based transmission, but not alpha-blended non-transmissive materials.

Note

If you want to enable refraction on your model, KHR_materials_transmission must also be used in addition with

KHR_materials_volume. See the dedicated *Volume* part of the documentation.

Warning

Transmission is complex for real-time rendering engines to implement, and support for the KHR_materials_transmission glTF extension is not yet widespread.

IOR

At import, there are two different situation:

- if KHR_materials_ior is not set, IOR value of Principled BSDF node is set to 1.5, that is the glTF default value of IOR.
- If set, the KHR_materials_ior is used to set the IOR value of Principled BSDF.

At export, IOR is included in the export only if one of these extensions are also used:

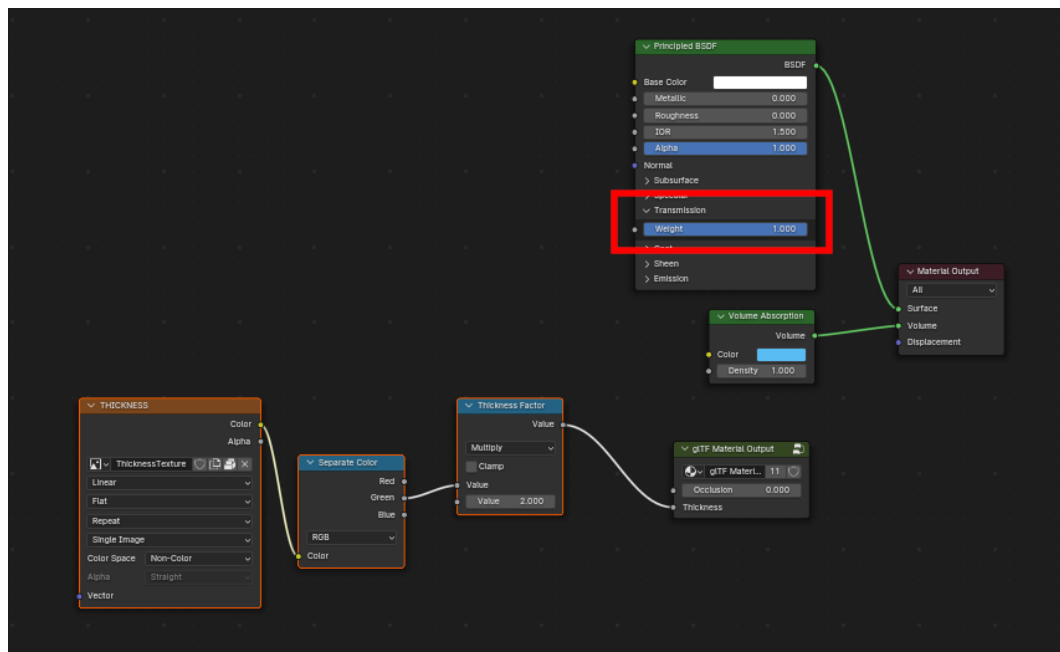
- KHR_materials_transmission
- KHR_materials_volume
- KHR_materials_specular

IOR of 1.5 are not included in the export, because this is the default glTF IOR value.

Volume

Volume can be exported using a Volume Absorption node, linked to Volume socket of Output node. Data will be exported using the KHR_materials_volume extension.

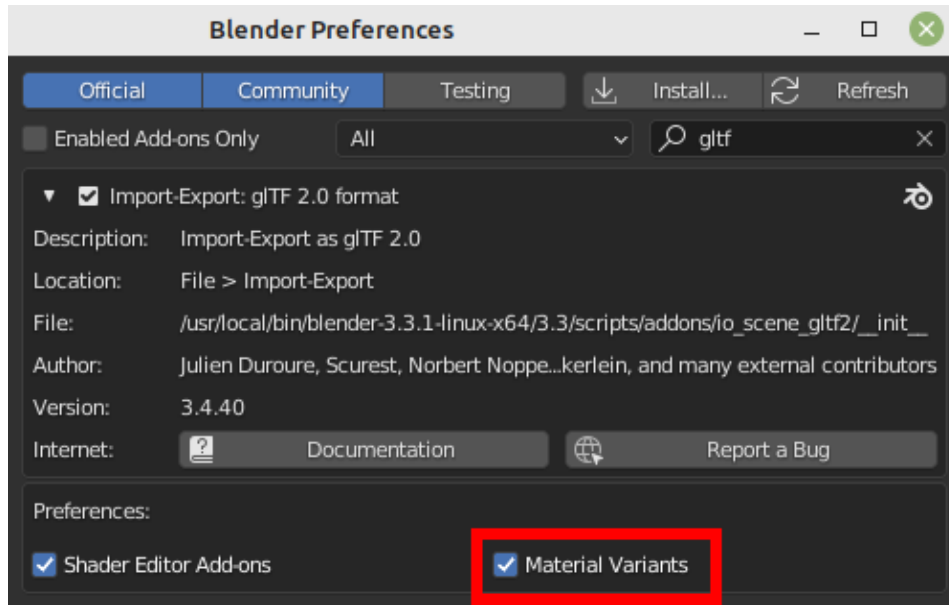
- For volume to be exported, some *transmission* must be set on Principled BSDF node.
- Color of Volume Absorption node is used as glTF attenuation color. No texture is allowed for this property.
- Density of Volume Absorption node is used as inverse of glTF attenuation distance.
- Thickness can be plugged into the Thickness socket of custom group node glTF Material Output.
- If a texture is used for thickness, it must be plugged on (G) Green channel of the image.



glTF Variants

Note

For a full Variants experience, you have to enable UI in Add-on preferences



There are two location to manage glTF Variants in Blender

- In 3D View, on glTF Variants tab
- For advanced settings, in Mesh Material Properties (see Advanced glTF Variant checks)

The main concept to understand for using Variants, is that each material slot will be used as equivalent of a glTF primitive.

glTF Variants switching

After importing a glTF file including KHR_materials_variants extension, all variants can be displayed.



You can switch Variant, by *selecting* the variant you want to display, then clicking on *Display Variant*.

You can switch to default materials (when no Variant are used), by clicking on *Reset to default*.

glTF Variants creation

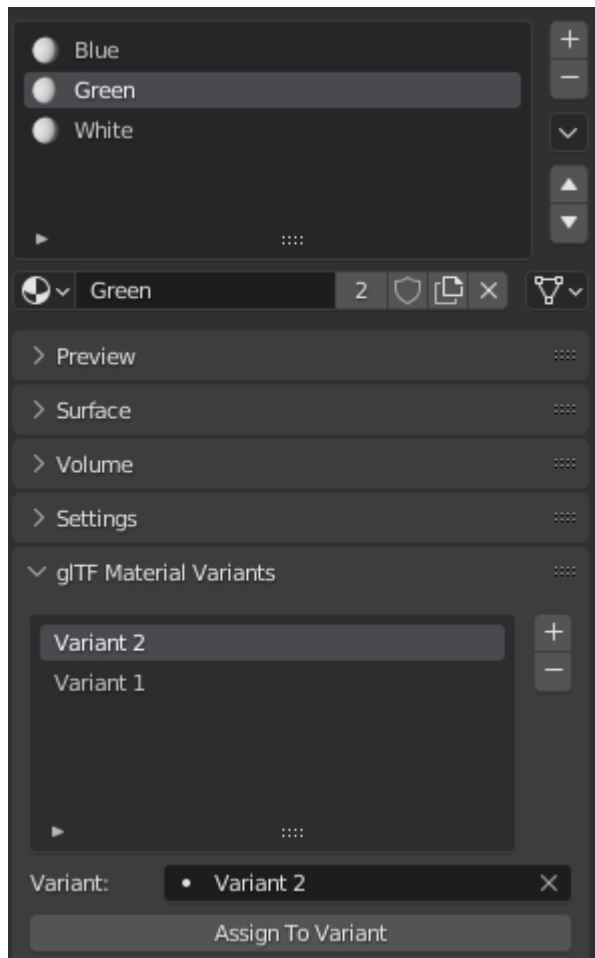
You can add a new Variant by clicking the + at right of the Variant list. Then you can change the name by double-clicking.

After changing Materials in Material Slots, you can assign current materials to the active Variant using *Assign to Variant*.

You can also set default materials using *Assign as Original*. These materials will be exported as default material in glTF. These are materials that will be displayed by any viewer that doesn't manage KHR_materials_variants extension.

Advanced glTF Variant checks

If you want to check primitive by primitive, what are Variants used, you can go to Mesh Material Properties.



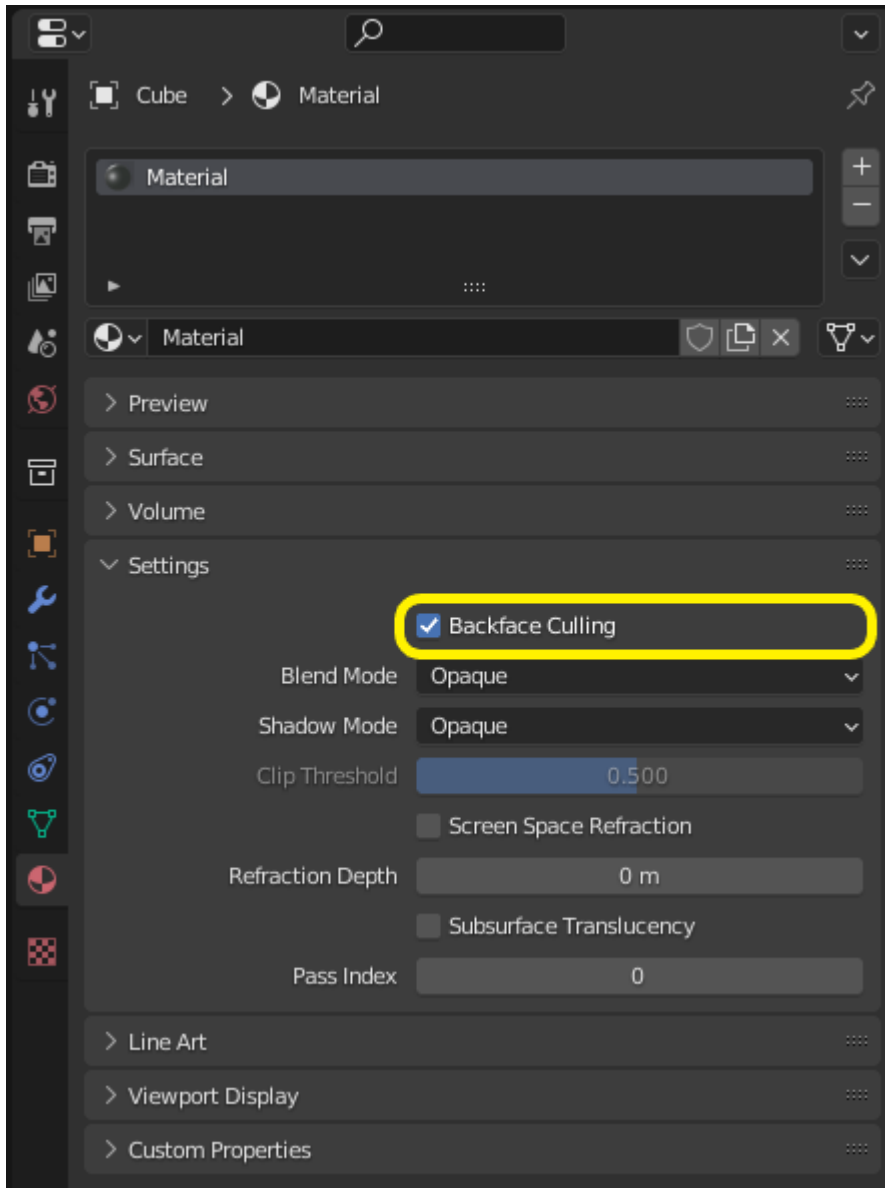
The *glTF Material Variants* tab refers to the active material Slot and Material used by this slot. You can see every Variants that are using this material for the given Slot/Primitive.

You can also assign material to Variants from this tab, but recommendation is to perform it from 3D View tab.

Double-Sided / Backface Culling

For materials where only the front faces will be visible, turn on *Backface Culling* in the *Settings* panel of an EEVEE material. When using other engines (Cycles, Workbench) you can temporarily switch to EEVEE to configure this setting, then switch back.

Leave this box unchecked for double-sided materials.



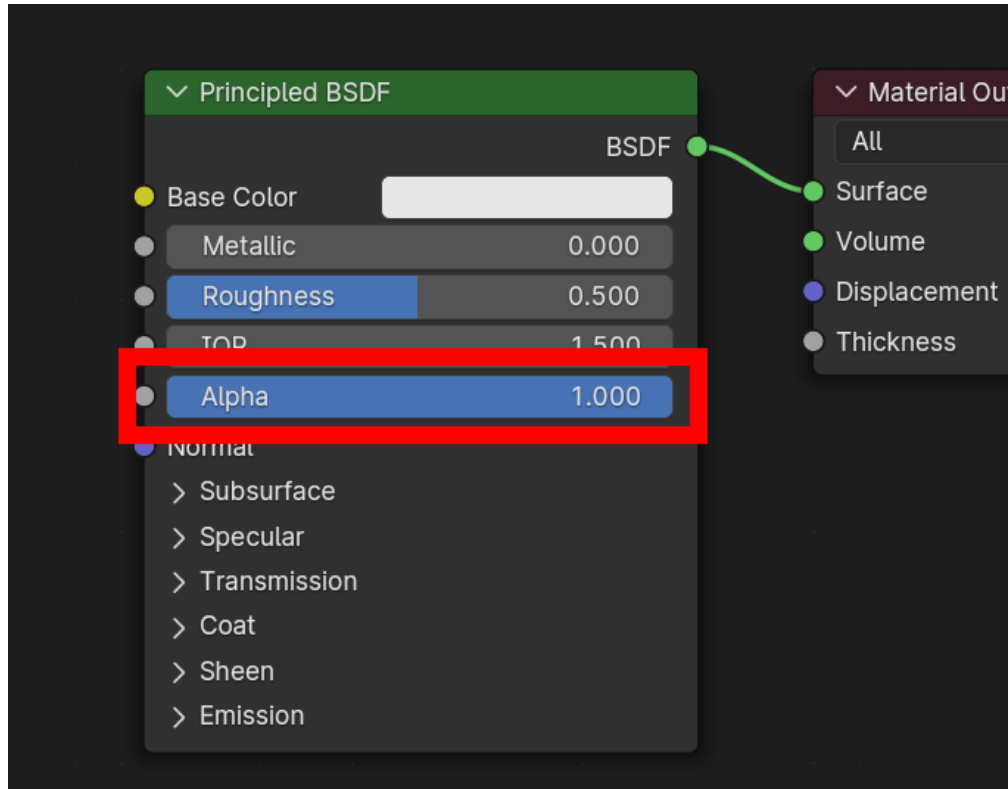
The inverse of this setting controls glTF's DoubleSided flag.

Alpha Modes

glTF has three alpha modes, depending on whether the alpha value is always 1, always 0 or 1, or can be between 0 and 1. The exporter determines the alpha mode automatically from the nodes connected to the Alpha socket.

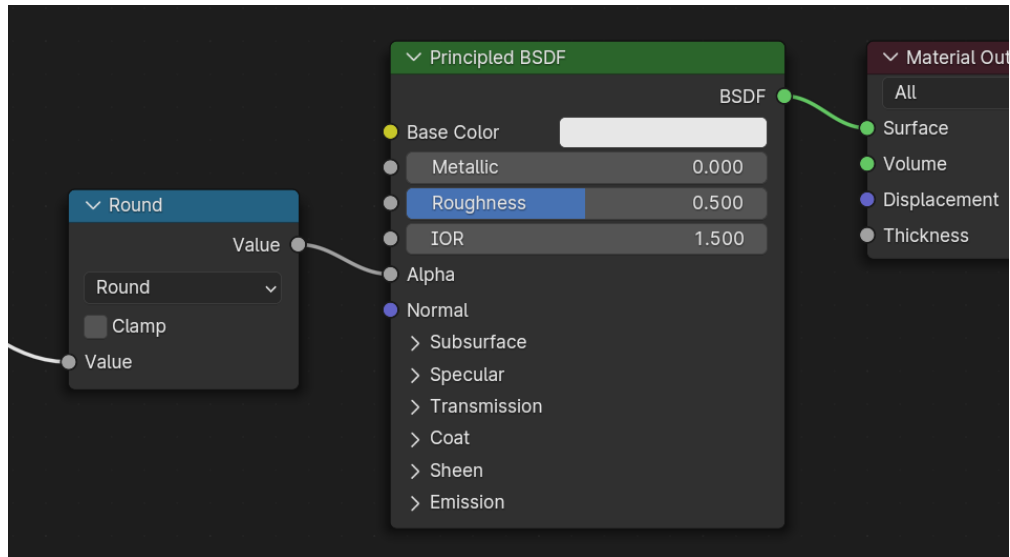
Opaque

In *opaque mode*, the material alpha is always 1.



Mask

In *mask mode*, the material alpha is always 0 or 1. This creates “cutout” transparency, where there is a hard edge between opaque and transparent regions, and can be used for things like leaves or cloth with holes. To enable this mode, use a Math node to round the alpha value to either 0 or 1.



Rounding snaps alpha values that are 0.5 or greater up to 1, and ones below 0.5 down to 0. It is also possible to use a cutoff value different than 0.5 by using Math nodes to do $1 - (\text{alpha} < \text{cutoff})$.

Mask mode is essentially the same as EEVEE's "Alpha Clip" blend mode, but is done with shader nodes so it works in other render engines.

Blend

Materials that use neither of these will use *blend mode*. Blend mode allows partially transparent surfaces that tint the objects seen through them, like layers of colored film. However, partial transparency is complex to render, and glTF viewers may show visual artifacts in non-trivial scenes that use blend mode.

To avoid artifacts, it may be a good idea to separate out parts of a model that can use opaque or mask mode, and use blend mode only on the parts where it is necessary, or to use only a *single* layer of transparent polygons in front of opaque objects.

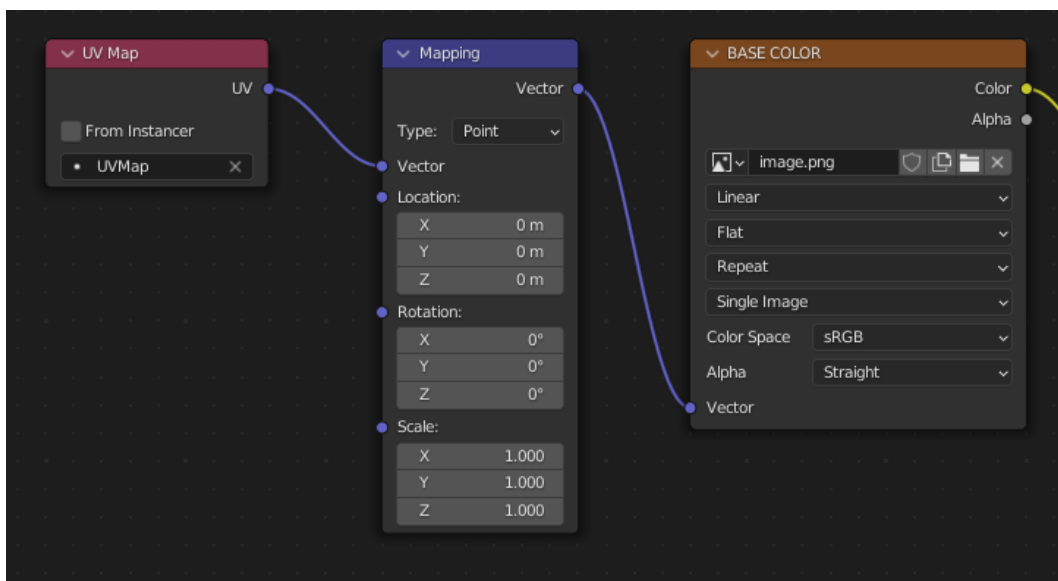
UV Mapping

Control over UV map selection and transformations is available by connecting a UV Map node and a Mapping node to any Image Texture node.

Settings from the Mapping node are exported using a glTF extension named KHR_texture_transform. There is a mapping type selector across the top. *Point* is the recommended type for export. *Texture* and *Vector* are also supported. The supported offsets are:

- *Location* - X and Y
- *Rotation* - Z only
- *Scale* - X and Y

For the *Texture* type, *Scale* X and Y must be equal (uniform scaling).



A deliberate choice of UV mapping.

Tip

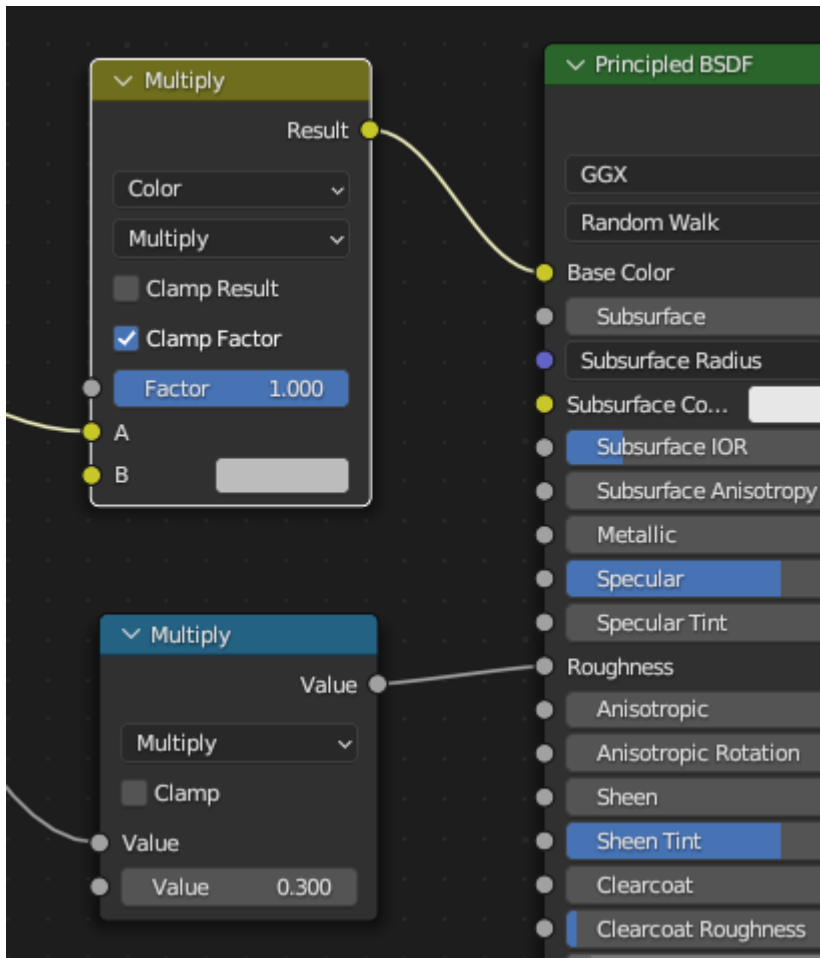
These nodes are optional. Not all glTF readers support multiple UV maps or texture transforms.

Factors

Any Image Texture nodes may optionally be multiplied with a constant color or scalar. These will be written as factors in the glTF file, which are

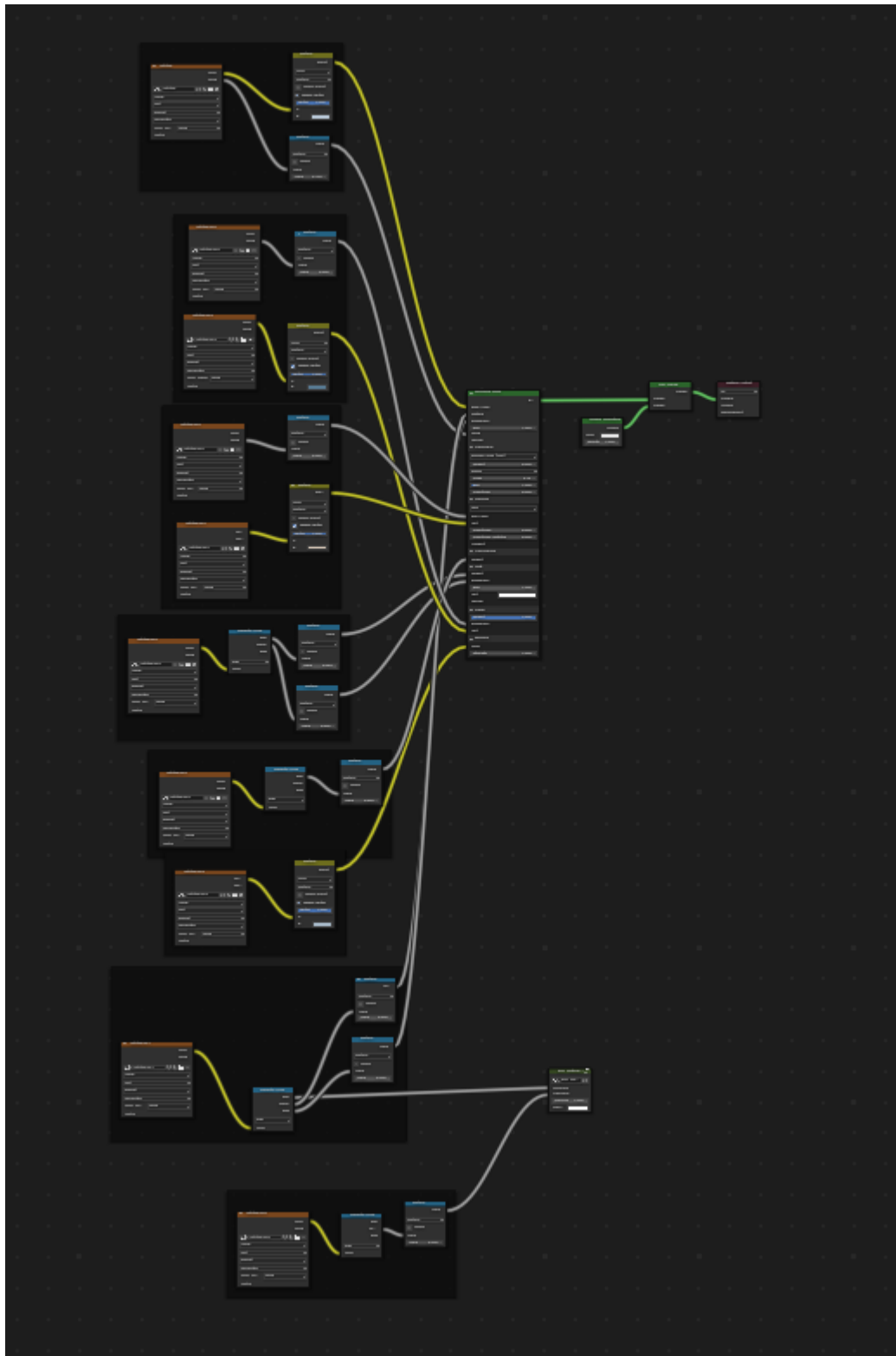
numbers that are multiplied with the specified image textures. These are not common.

- Use Math node (multiply) for scalar factors. Use second value as factor
- Use Mix node (color / multiply) for color factors. Set Factor to 1, and use Color2 (B) as factors



Example

A single material may use all of the above at the same time, if desired. This figure shows a typical node structure when several of the above options are applied at once:



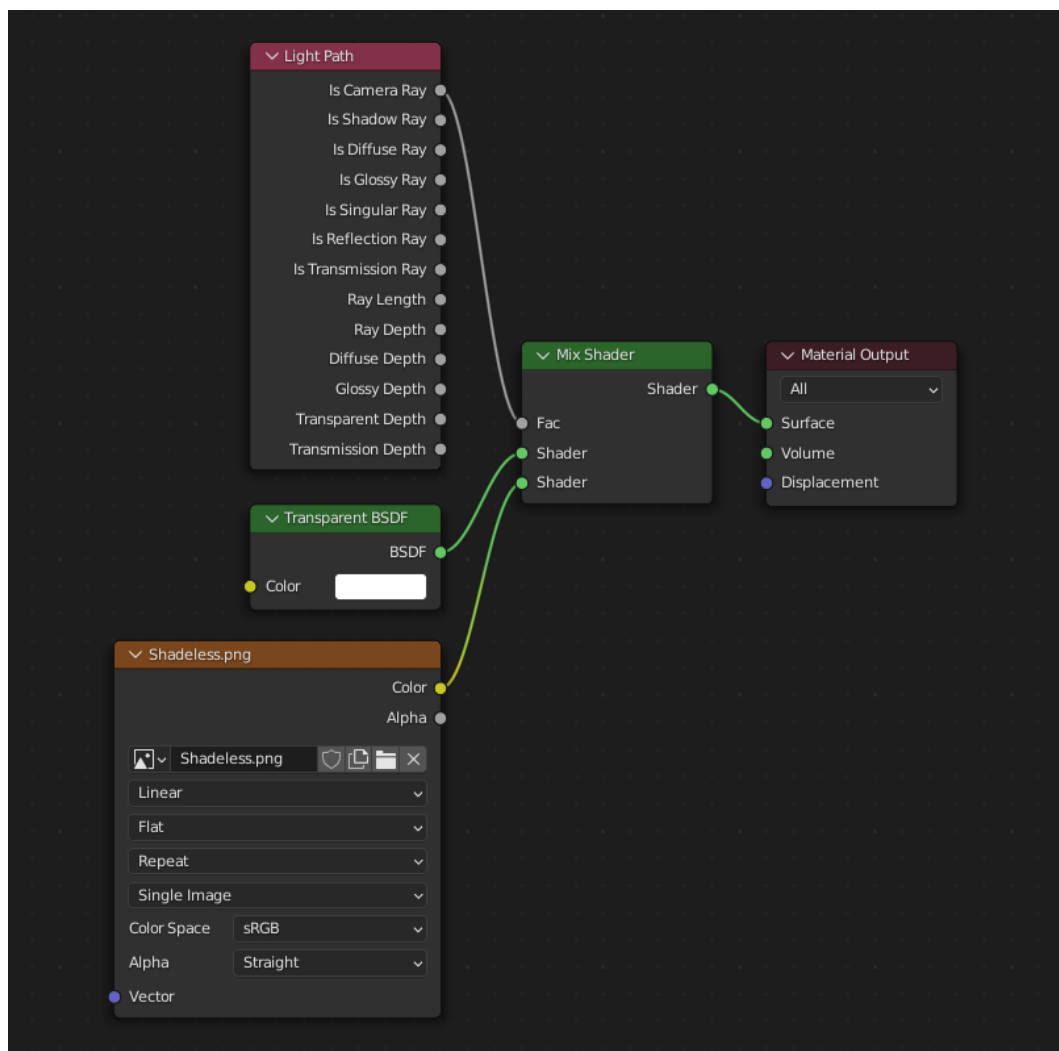
A Principled BSDF material with an emissive texture.

UDIM

UDIM is a way to store multiple textures in a single image file. The UDIM system is supported by Blender, but is not supported by glTF. When exporting a model that uses UDIM, the add-on will automatically split the image into multiple images, one for each tile, and will update the material nodes to use the new images. All UDIM texture must use the same UV map to be exported.

Exporting a Shadeless (Unlit) Material

To export an unlit material, mix in a camera ray, and avoid using the Principled BSDF node.



One of several similar node arrangements that will export KHR_materials_unlit and render shadeless in Blender.

Extensions

The core glTF 2.0 format can be extended with extra information, using glTF extensions. This allows the file format to hold details that were not considered universal at the time of first publication. Not all glTF readers support all extensions, but some are fairly common.

Certain Blender features can only be exported to glTF via these extensions. The following [glTF 2.0 extensions](#) are supported directly by this add-on:

Import

- KHR_materials_pbrSpecularGlossiness
- KHR_materials_clearcoat
- KHR_materials_transmission
- KHR_materials_unlit
- KHR_materials_emissive_strength
- KHR_materials_volume
- KHR_materials_sheen
- KHR_materials_specular
- KHR_materials_anisotropy
- KHR_materials_ior
- KHR_materials_variants
- KHR_lights_punctual
- KHR_texture_transform
- KHR_mesh_quantization
- EXT_mesh_gpu_instancing

Export

- KHR_draco_mesh_compression
- KHR_lights_punctual
- KHR_materials_clearcoat

- KHR_materials_transmission
- KHR_materials_unlit
- KHR_materials_emissive_strength
- KHR_materials_volume
- KHR_materials_sheen
- KHR_materials_specular
- KHR_materials_anisotropy
- KHR_materials_ior
- KHR_materials_variants
- KHR_texture_transform
- EXT_mesh_gpu_instancing

Third-party glTF Extensions

It is possible for Python developers to add Blender support for additional glTF extensions by writing their own third-party add-on, without modifying this glTF add-on. For more information, [see the example on GitHub](#) and if needed, [register an extension prefix](#).

Custom Properties

Custom properties are always imported, and will be exported from most objects if the *Include ▶ Custom Properties* option is selected before export. These are stored in the `extras` field on the corresponding object in the glTF file.

Unlike glTF extensions, custom properties (extras) have no defined namespace, and may be used for any user-specific or application-specific purposes.

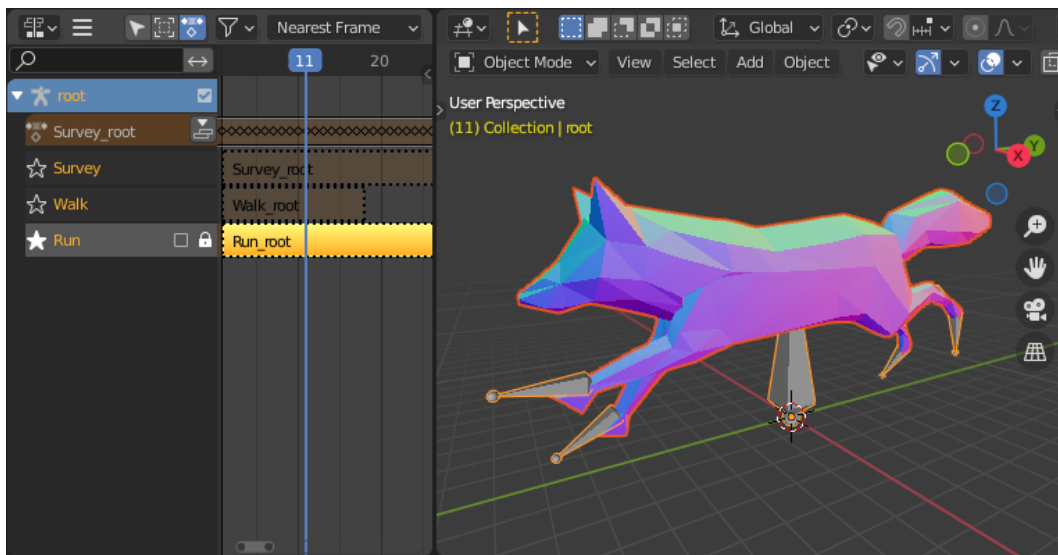
Animations

A glTF animation changes the transforms of objects or pose bones, or the values of shape keys. One animation can affect multiple objects, and there can be multiple animations in a glTF file.

Import

Imported models are set up so that the first animation in the file is playing automatically. Scrub the Timeline to see it play.

When the file contains multiple animations, the rest will be organized using the [Nonlinear Animation editor](#). Each animation becomes an action stashed to an NLA track. The track name is the name of the glTF animation. To make the animation within that track visible, click Solo (star icon) next to the track you want to play.



This is the [fox sample model](#) showing its “Run” animation.

If an animation affects multiple objects, it will be broken up into multiple parts. The part of the animation that affects one object becomes an action stashed on that object. Use the track names to tell which actions are part of the same animation. To play the whole animation, you need to enable Solo (star icon) for all its tracks.

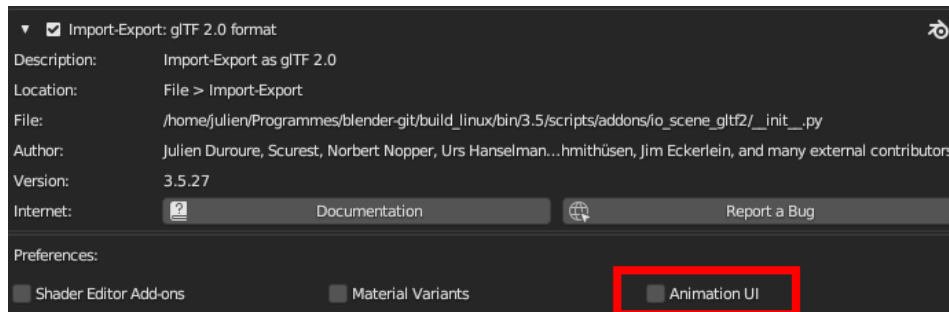
Note

There is currently no way to see the non-animated pose of a model that had animations.

You can also use the animation switcher that can be found in [Dope Sheet editor](#).

Note

You have to enable UI in Add-on preferences for seeing the animation switcher



You can switch all animation imported. It automatically enables Solo (star icon) for all needed tracks. It also reset non animated object to Rest transformation.

Export

You can export animations using different ways. How glTF animations are made from actions / NLA is controlled by the *Animation ▶ Mode* export option.

Actions (default)

An action will be exported if it is the active action on an object, or it is stashed to an NLA track (e.g. with the *Stash* or *Push Down* buttons in the [Action Editor](#)). Actions which are **not** associated with an object in one of these ways are **not exported**. If you have multiple actions you want to export, make sure they are stashed!

A glTF animation can have a name, which is the action name by default. You can override it by renaming its NLA track from NLATrack/[Action

Stash] to the name you want to use. For example, the Fig. [fox model](#) will export with three animations, “Survey”, “Walk”, and “Run”. If you rename two tracks on two different objects to the same name, they will become part of the same glTF animation and will play together.

The importer organizes actions so they will be exported correctly with this mode.

This mode is useful if you are exporting for game engine, with an animation library of a character. Each action must be on its own NLA track.

Active Actions merged

In this mode, the NLA organization is not used, and only one animation is exported using the active actions on all objects.

NLA Tracks

In this mode, each NLA Track will be export as an independent glTF animation. This mode is useful if you are using Strip modifiers, or if you get multiple action on a same Track.

If you rename two tracks on two different objects to the same name, they will become part of the same glTF animation and will play together.

Scene

Using [Scene](#) option, animations will be exported as you can see them in viewport. You can choose to export a single glTF animation, or each object separately.

Note

Remember only certain types of animation are supported:

- Object transform (location, rotation, scale)
- Pose bones

- Shape key values

Animation of other properties, like physics, lights, or materials, will be ignored.

Note

In order to sample shape key animations controlled by drivers using bone transformations, they must be on a mesh object that is a direct child of the bones' armature.

Note

Only [Actions \(default\)](#) and [Active Actions merged](#) mode can handle not sampled animations.

File Format Variations

The glTF specification identifies different ways the data can be stored. The importer handles all of these ways. The exporter will ask the user to select one of the following forms:

glTF Binary (.glb)

This produces a single .glb file with all mesh data, image textures, and related information packed into a single binary file.

Tip

Using a single file makes it easy to share or copy the model to other systems and services.

glTF Separate (.gltf + .bin + textures)

This produces a JSON text-based `.gltf` file describing the overall structure, along with a `.bin` file containing mesh and vector data, and optionally a number of `.png` or `.jpg` files containing image textures referenced by the `.gltf` file.

Tip

Having an assortment of separate files makes it much easier for a user to go back and edit any JSON or images after the export has completed.

Note

Be aware that sharing this format requires sharing all of these separate files together as a group.

glTF Embedded (.gltf)

This produces a JSON text-based `.gltf` file, with all mesh data and image data encoded (using Base64) within the file. This form is useful if the asset must be shared over a plain-text-only connection.

Warning

This is the least efficient of the available forms, and should only be used when required. Available only when you activated it in addon preferences.

Properties

Import

Pack Images

Pack all images into the blend-file.

Merge Vertices

The glTF format requires discontinuous normals, UVs, and other vertex attributes to be stored as separate vertices, as required for rendering on typical graphics hardware. This option attempts to combine co-located vertices where possible. Currently cannot combine verts with different normals.

Shading

How normals are computed during import.

Guess Original Bind Pose

Determines the pose for bones (and consequently, skinned meshes) in Edit Mode. When on, attempts to guess the pose that was used to compute the inverse bind matrices.

Bone Direction

Changes the heuristic the importer uses to decide where to place bone tips. Note that the Fortune setting may cause inaccuracies in models that use non-uniform scaling. Otherwise this is purely aesthetic. The default value will not change axis, and is best for re-exporting from Blender. This default option will change display mode (adding shape and changing relationship line) to have a better view, even if original bones axis are not the most accurate (esthetically speaking)

Lighting Mode

Optional backwards compatibility for non-standard render engines. Applies to lights. Standard: Physically-based glTF lighting units (cd, lx, nt). Unitless: Non-physical, unitless lighting. Useful when exposure controls are not available Raw (Deprecated): Blender lighting strengths with no conversion

Import WebP textures

If a texture exists in WebP format, loads the WebP texture instead of the fallback png/jpg one.

Export

Format

See: [File Format Variations](#).

Keep Original

For glTF Separate file format only. Keep original textures files if possible. Warning: if you use more than one texture, where PBR standard requires only one, only one texture will be used. This can lead to unexpected results

Textures

For glTF Separate file format only. Folder to place texture files in. Relative to the gltf-file.

Copyright

Legal rights and conditions for the model.

Remember Export Settings

Store export settings in the blend-file, so they will be recalled next time the file is opened.

Include

Selected Objects

Export selected objects only.

Visible Objects

Export visible objects only.

Renderable Objects

Export renderable objects only.

Active Collection

Export objects from active collection only.

Include Nested Collections

Only when Active Collection is On. When On, export recursively objects on nested active collections.

Active Scene

Export active scene only.

Custom Properties

Export custom properties as glTF extras.

Cameras

Export cameras.

Punctual Lights

Export directional, point, and spot lights. Uses the KHR_lights_punctual glTF extension.

Transform

Y Up

Export using glTF convention, +Y up.

Data - Scene Graph

Geometry Nodes Instances

Export Geometry nodes instances. This feature is experimental.

GPU Instances

Export using EXT_mesh_gpu_instancing extensions.

Flatten Object Hierarchy

Useful in case of non-decomposable TRS matrix. Only skinned meshes will stay children of armature.

Full Collection Hierarchy

Export collections as empty, keeping full hierarchy. If an object is in multiple collections, it will be exported it only once, in the first collection it is found.

Data - Mesh

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all [Modifiers](#) have been calculated.

UVs

Export UVs (texture coordinates) with meshes.

Normals

Export vertex normals with meshes.

Tangents

Export vertex tangents with meshes.

Attributes

Export Attributes with meshes, when the name starts with underscore.

Loose Edges

Export loose edges as lines, using the material from the first material slot.

Loose Points

Export loose points as glTF points, using the material from the first material slot.

Shared Accessor

For triangles, use shared accessor for indices. This is more efficient (smaller files when you have lots of materials).

Data - Mesh - Vertex Color

Use Vertex Color

Material:: Export vertex color when used in material node tree as Base Color multiplier. This is the default, and the most accurate regarding glTF specification.

Active:: Export active vertex colors, even if not used in material node tree. A fully compliant glTF viewer should display this VC as Base Color multiplier.

None:: Do not export vertex color.

Export all vertex colors

Export all vertex colors, additional VC will be COLOR_1, COLOR_2, etc.

Export active vertex color when no material

Export active vertex color when no material is assigned to the object.

Data - Material

Materials

Export full materials, only placeholders (all primitives but without materials), or does not export materials. (In that last case, primitives are merged, losing material slot information).

Images

Output format for images. PNG is lossless and generally preferred, but JPEG might be preferable for web applications due to the smaller file size. If WebP is chosen, all textures will be saved as WebP, without any png/jpg fallback. If None is chosen, materials are exported without textures.

Image Quality

When exporting jpeg or WebP files, the quality of the exported file.

Create WebP

Creates WebP textures for every textures, in addition to the existing texture. For already WebP textures, nothing happen.

WebP fallback

For all WebP textures, create a png fallback texture.

Unused images

Export images that are not used in any material.

Unused textures

Export texture info (sampler, image, texcoord) that are not used in any material.

Data - Shape Keys

Export shape keys (morph targets).

Shape Key Normals

Export vertex normals with shape keys (morph targets).

Shape Key Tangents

Export vertex tangents with shape keys (morph targets).

Data - Shape Keys - Optimize

Use Sparse Accessor if better

Sparse Accessor will be used if it save space (if the exported file is smaller)

Omitting Sparse Accessor if data is empty

If data is empty, omit to export SPArce Accessor. Not all viewer managed it correctly, so this option is Off by default

Data - Armature

Use Rest Position Armature

Export Armatures using rest position as joint rest pose. When Off, the current frame pose is used as rest pose.

Export Deformation Bones only

Export Deformation bones only, not other bones. Animation for deformation bones are baked.

Remove Armature Object

Remove Armature Objects if possible. If some armature(s) have multiple root bones, we can't remove them.

Flatten Bone Hierarchy

Useful in case of non-decomposable TRS matrix.

Data - Skinning

Export skinning data

Bone influences

How many joint vertex influences will be exported. Models may appear incorrectly in many viewers with value different to 4 or 8.

Include All Bone Influences

Export all joint vertex influences. Models may appear incorrectly in many viewers.

Data - Lighting

Lighting Mode

Optional backwards compatibility for non-standard render engines. Applies to lights. Standard: Physically-based glTF lighting units (cd, lx, nt). Unitless: Non-physical, unitless lighting. Useful when exposure controls are not available Raw (Deprecated): Blender lighting strengths with no conversion

Data - Compression

Compress meshes using Google Draco.

Compression Level

Higher compression results in slower encoding and decoding.

Quantization Position

Higher values result in better compression rates.

Normal

Higher values result in better compression rates.

Texture Coordinates

Higher values result in better compression rates.

Color

Higher values result in better compression rates.

Generic

Higher values result in better compression rates.

Animation

Animation mode

Animation mode used for export (See [Animations](#))

Shape Keys Animations

Export Shape Keys Animation. Need Shape Keys to be exported (See [Data - Shape Keys](#))

Bake All Objects Animations

Useful when some objects are constrained without being animated themselves.

Animation - Rest & Ranges

Use Current Frame as Object Rest Transformations

Export the scene in the current animation frame. When off, frame 0 is used as rest transformation for objects.

Limit to Playback Range

Clips animations to selected playback range.

Set all glTF Animation starting at 0

Set all glTF Animation starting at 0. Can be useful for looping animation

Negative Frames

When some frames are in negative range, slide or crop the animation.

Animation - Armature

Export all Armature Actions

Export all actions, bound to a single armature. Warning: Option does not support exports including multiple armatures.

Reset pose bones between actions

Reset pose bones between each action exported. This is needed when some bones are not keyed on some animations.

Animation - Sampling

Apply sampling to all animations. Do not sample animation can lead to wrong animation export.

Sampling Rate

How often to evaluate animated values (in frames).

Animation - Optimize

Optimize Animation Size

Reduce exported file size by removing duplicate keyframes when all identical.

Force keeping channel for armature / bones

if all keyframes are identical in a rig, force keeping the minimal animation.

Force keeping channel for objects

if all keyframes are identical for object transformations, force keeping the minimal animation.

Disable viewport for other objects

When exporting animations, disable viewport for other objects, for performance reasons, when possible.

Animation - Filter

Restrict actions to be exported to the ones matching the filter.

Contributing

This importer/exporter is developed through the [glTF-Blender-IO repository](#), where you can file bug reports, submit feature requests, or contribute code.

Discussion and development of the glTF 2.0 format itself takes place on the Khronos Group [glTF GitHub repository](#), and feedback there is welcome.

Node

These add-ons relate to the node editors and related tools.

- [Node Wrangler](#)

Node Wrangler

Node Wrangler provides various tools that help you to work with nodes quickly and efficiently.

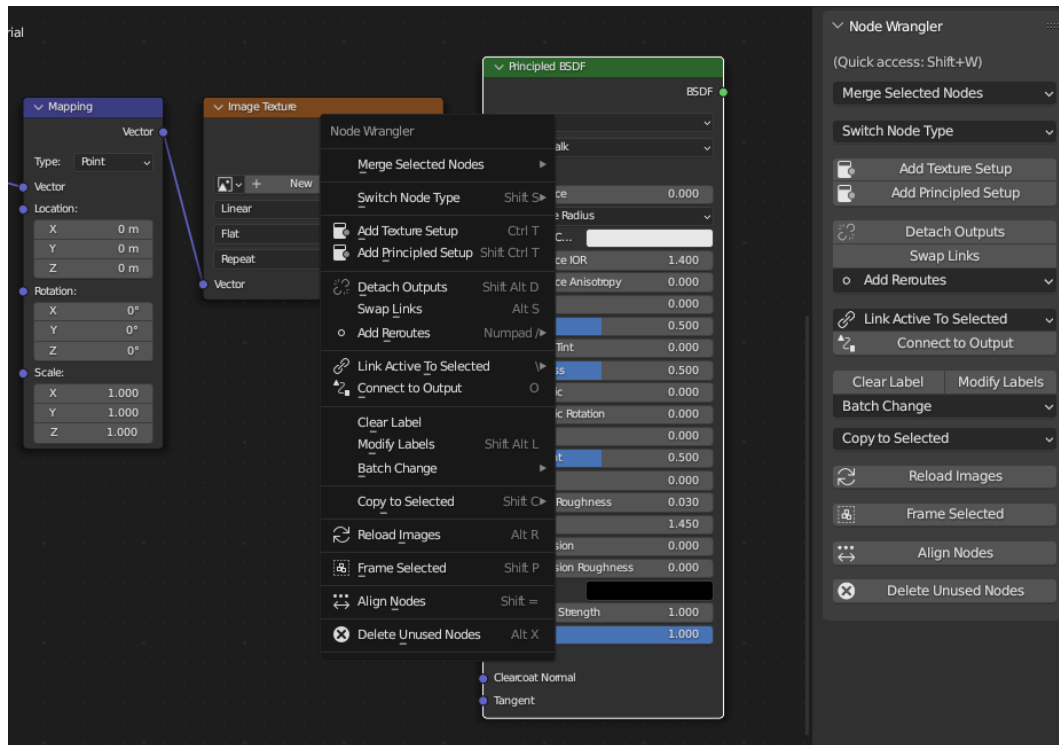
While many of this add-on's functions work in all supported node editors (Compositor, Shader, Geometry Nodes, and Texture Nodes) some functions only work in specific node editors, and some functions work differently per editor. Functions that only work in specific editors are marked with labels (*Compositor*, *Shader*, *Geometry Nodes*, *Texture Nodes*). Functions without labels should work for all node editors.

Enabling Add-on

1. Open Blender and go to [Add-ons](#) section of the [Preferences](#).
2. Search “Node Wrangler” and check the *Enable Extension* checkbox.

Usage

Use the panel in Sidebar of the node editor or press Shift-W to bring up the quick access menu. You can also look up the shortcut list in the add-on preferences panel.



You can access most functions from the sidebar panel or quick access menu.

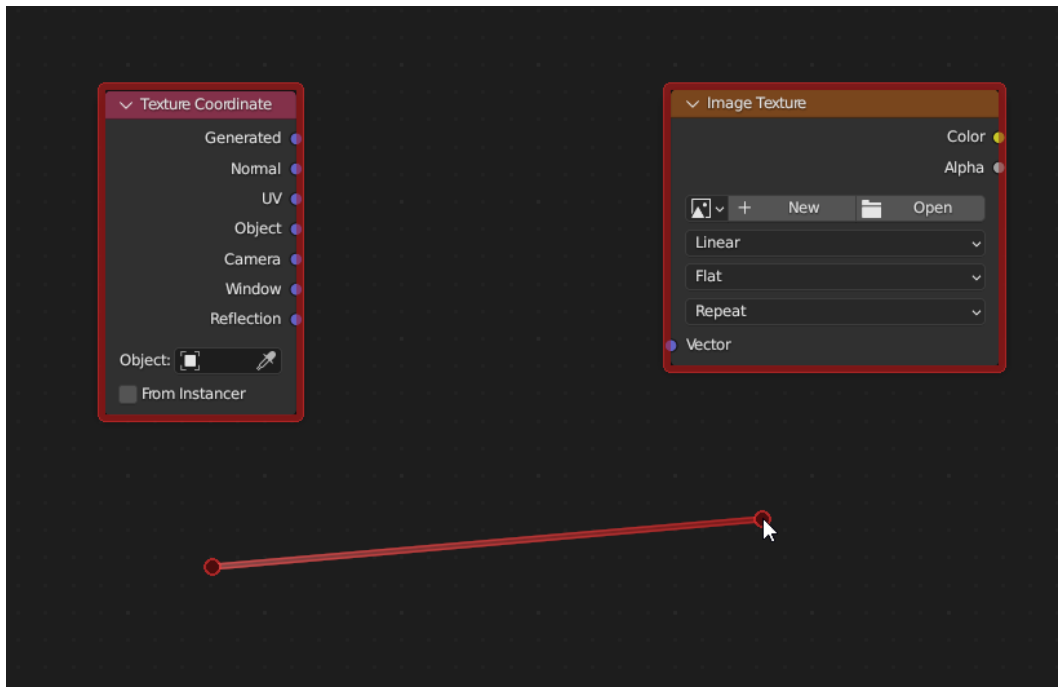
Description

Lazy Connect

Reference

Shortcut:: Alt-RMB-drag, Shift-Alt-RMB-drag

Connect two nodes without even clicking the sockets. Just drag the cursor from one node to another while holding Alt-RMB. It will select the nodes nearest the start and end points of the drag for connection, so you don't even have to click on the nodes.



Selection can be lazy.

It tries to connect the best-matched sockets possible, based on their names, types, and whether they are open or not.

For a more precise connection, you can alternatively use Shift-Alt-RMB. It brings up menus of available inputs and outputs before connection, so you can select the exact sockets to connect. It's especially useful when working with a large node tree since you can make connections without frequently zooming in and out.

Lazy Mix

Reference

Shortcut:: Shift-Ctrl-RMB-drag

Connect the outputs of two nodes into an appropriate “mix” type of node. This is the “lazy” way of selecting nodes and executing the *Mix* function from [Merge with Automatic Type Detection](#).

Merge

Reference

Menu:: *Node Wrangler ▶ Merge Selected Nodes*

Connect outputs of the selected nodes into a “mix” type of node (Mix, Math, Z-Combine, Alpha Over, Mix Shader, Add Shader, Join Geometry).

Note

Merge currently does not support outputs of Integer, String, or Boolean types from Geometry Nodes.

There are automatic and manual ways of merging. The automatic ways let the add-on determine which “mix” node to use based on the types of outputs to merge. The manual ways let you decide and force connections even if the types of outputs and the “mix” node are not compatible.

Note

Generally, the modifier part of the shortcut signifies the type of “mix” node you want to use (Ctrl for automatic detection, Ctrl-Alt for the Mix node, and Shift-Ctrl for the Math node), the non-modifier part signifies the mode of “mix” node you want to set (NumpadPlus for add, NumpadMinus for subtract, NumpadSlash for divide, and NumpadAsterisk for multiply).

Merge with Automatic Type Detection

The automatic merge functions determine the type of “mix” node to use based on the types of outputs to merge. If it has a Color output, it will use the Mix node. It will use the Math node if both outputs are of Value type.

Add Shader, Mix Shader, and Join Geometry nodes will also be used for specific cases.

Modes

Add Ctrl=, Ctrl-NumpadPlus

Merge into Mix or Math nodes, then set blend mode or math operation as Add. If the outputs are Shaders, it will use Add Shader node instead.

Multiply Ctrl-8, Ctrl-NumpadAsterisk

Merge into Mix or Math nodes, then set blend mode or math operation as Multiply.

Subtract Ctrl-Minus, Ctrl-NumpadMinus

Merge into Mix or Math nodes, then set blend mode or math operation as Subtract.

Divide Ctrl-Slash, Ctrl-NumpadSlash

Merge into Mix or Math nodes, then set blend mode or math operation as Divide.

Mix Ctrl-0, Ctrl-Numpad0

Merge into Mix node, then set blend mode as Mix. If the outputs are Shaders, it will use Mix Shader node instead. If the outputs are Geometry, it will use Join Geometry node.

Merge Using Mix Node

Reference

Menu:: *Node Wrangler ▶ Merge Selected Nodes ▶ Use Mix Nodes*

Use the Mix nodes for merging, regardless of the selected nodes. You can choose the mode of the node via the menu. You can quickly set some operations by using corresponding shortcuts.

- Add: Ctrl-Alt-=, Ctrl-Alt-=
- Subtract: Ctrl-Alt-Minus, Ctrl-Alt-NumpadMinus
- Multiply: Ctrl-Alt-8, Ctrl-Alt-NumpadAsterisk
- Divide: Ctrl-Alt-Slash, Ctrl-Alt-NumpadSlash

Merge Using Math Node

Reference

Menu:: *Node Wrangler ▶ Merge Selected Nodes ▶ Use Math Nodes*

Use the Math nodes for merging, regardless of the selected nodes. You can choose the mode of the node via the menu. You can quickly set some operations by using corresponding shortcuts.

- Add: Shift-Ctrl-=, Shift-Ctrl-=
- Subtract: Shift-Ctrl-Minus, Shift-Ctrl-NumpadMinus
- Multiply: Shift-Ctrl-8, Shift-Ctrl-NumpadAsterisk
- Divide: Shift-Ctrl-Slash, Shift-Ctrl-NumpadSlash
- Greater than: Ctrl-Comma
- Less than: Ctrl-Period

Merge Using Z-Combine Node

Compositor

Reference

Menu:: *Node Wrangler ▶ Merge Selected Nodes ▶ Use Z-Combine Nodes*

Shortcut:: Ctrl-NumpadPeriod

Use the Z-Combine nodes for merging. If possible, Image and Z-Depth outputs will be linked. If the current node editor is not Compositor, this will execute the *Mix* function from the automatic merge.

Merge Using Alpha Over Node

Compositor

Reference

Menu:: *Node Wrangler ▶ Merge Selected Nodes ▶ Use Alpha Over Nodes*

Shortcut:: *Ctrl-Alt-0*

Use the Alpha Over nodes for merging. If the current node editor is not Compositor, this will execute the *Mix* function from the automatic merge.

Batch Change Blend Mode / Math Operation

Reference

Menu:: *Node Wrangler ▶ Batch Change*

Change the blend mode or math operation of the selected Mix and Math nodes at once. You can use Alt-Up or Alt-Down to cycle through previous or next blend modes or math operations. You can also quickly set some operations by using corresponding shortcuts.

- Add: Alt-=, Alt-=
- Subtract: Alt-Minus, Alt-NumpadMinus
- Multiply: Alt-8, Alt-NumpadAsterisk
- Divide: Alt-Slash, Alt-NumpadSlash
- Greater than: Alt-Comma
- Less than: Alt-Period

Change Mix Factor

Reference

Shortcut:: Alt-Left, Shift-Alt-Left, Alt-Right, Shift-Alt-Right, Shift-Ctrl-Alt-Left, Shift-Ctrl-Alt-0, Shift-Ctrl-Alt-Right, Shift-Ctrl-Alt-1

Change the Factor value of the selected Mix and Mix Shader nodes with shortcuts.

- Increase Factor by 0.1: Alt-Right
- Decrease Factor by 0.1: Alt-Left
- Increase Factor by 0.01: Shift-Alt-Right
- Decrease Factor by 0.01: Shift-Alt-Left
- Set Factor to 0.0: Shift-Ctrl-Alt-Left, Shift-Ctrl-Alt-0
- Set Factor to 1.0: Shift-Ctrl-Alt-Right, Shift-Ctrl-Alt-1

Delete Unused Nodes

Reference

Menu:: *Node Wrangler ▶ Delete Unused Nodes*
Shortcut:: Alt-X

Clean up your node tree. Delete all nodes that don't contribute to the final result.

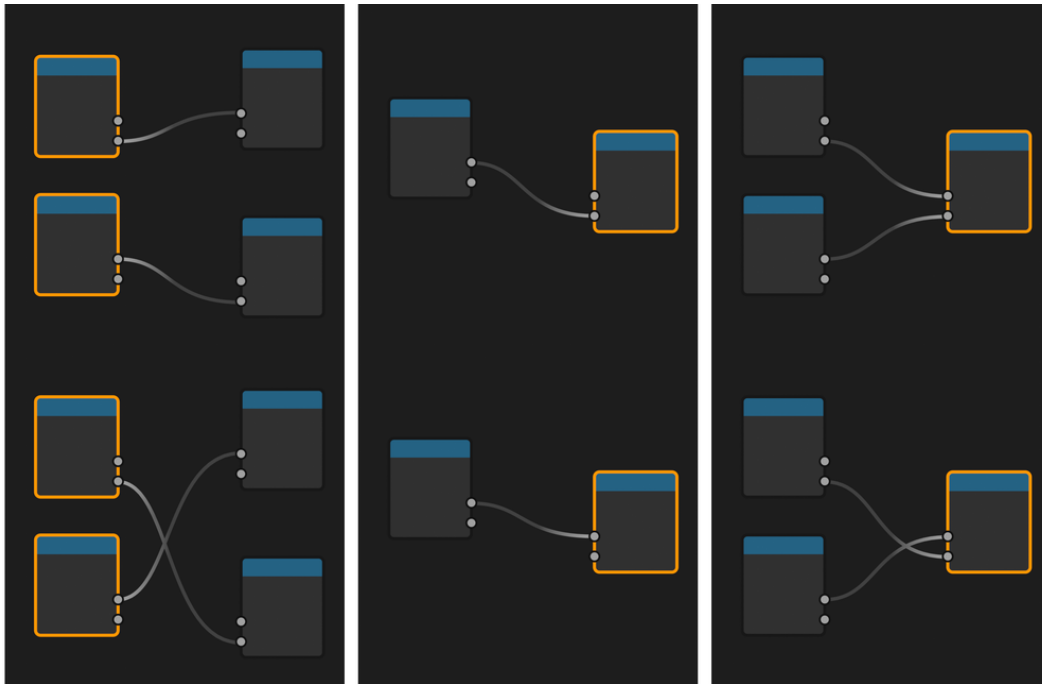
Swap Links

Reference

Menu:: *Node Wrangler ▶ Swap Links*
Shortcut:: Alt-S

When two nodes are selected, this swaps each other's output link. Note that some output connections can be lost if the two nodes have a different number of connected outputs.

With one node selected, if the node has one linked input, it cycles the link through the available input sockets. If the node has two linked inputs, it swaps those two links. If there are more than two inputs linked, it swaps the two inputs with matching types (the Mix node's two Color inputs, for example).



Swap works differently depending on the selected nodes and their links.

Reset Backdrop

Compositor

Reference

Menu:: *Node Wrangler ▶ Reset Backdrop*

Shortcut:: *Z*

Reset the position and scale of the backdrop.

Add Attribute Node

Shader

Reference

Menu:: *Header ▶ Add ▶ Input ▶ Attributes*

Add an Attribute node with the selected attribute.

Preview Node Output

Shader Geometry Nodes

Reference

Shortcut:: Shift-Ctrl-LMB for *Shader*, Shift-Alt-LMB for *Geometry Nodes*

Connect an output of the selected node to the final output of the node tree (the Material Output or World Output for Shader, the final Group Output for Geometry Nodes) to preview its output in the viewport. You can cycle through the available outputs by clicking it again while holding the modifier keys.

See also

While in Shader, any output can be connected to the final output, in Geometry Nodes, only Geometry outputs can be connected to the final output. To preview other types of outputs in Geometry Nodes, use its own [Viewer Node](#).

See also

Also check out *Connect to Output*. It is a similar function but has different behaviors. It also works in all node editors.

Frame Selected

Reference

Menu:: *Node Wrangler ▶ Frame Selected*
Shortcut:: Shift-P

Insert the selected nodes into a Frame node.

Reload Images

Compositor Shader Texture Nodes

Reference

Menu:: *Node Wrangler ▶ Reload Images*
Shortcut:: Alt-R

Reload all of the images used in the node tree. This lets you reload the images without using the Image Editor.

Copy Settings

Reference

Menu:: *Node Wrangler ▶ Copy to Selected ▶ Settings from Active*
Shortcut:: Shift-C

Copy the settings of the active node to all selected nodes of the same type.

Reset Nodes

Reference

Shortcut:: Backspace

Revert the settings of the selected nodes to default while maintaining connections.

Copy Label

Reference

Menu:: *Node Wrangler ▶ Copy to Selected ▶ Copy Label*

Shortcut:: Shift-V, Shift-C

Copy custom labels to all of the selected nodes. You can copy them from the active node (Shift-V), from the nodes that are linked to the selected ones, or from the names of the sockets that the selected nodes are linked to. Shift-C will bring up a submenu with all available options.

Clear Label

Reference

Menu:: *Node Wrangler ▶ Clear Label*

Shortcut:: Alt-L

Clear the custom labels of selected nodes and revert them back to their default node names.

Modify Labels

Reference

Menu:: *Node Wrangler ▶ Modify Labels*

Shortcut:: Shift-Alt-L

Batch rename the custom labels of selected nodes. You can add text to the beginning and the end and replace parts of the text.

Add Texture Setup

Shader

Reference

Menu:: *Node Wrangler ▶ Add Texture Setup*
Shortcut:: Ctrl-T

Add a setup of a texture node, Texture Coordinate, and Mapping nodes to any shader node. If you select a texture node, it will only add the Texture Coordinate and Mapping nodes. For a background shader it will add an Environment Texture node.

Add Principled Texture Setup

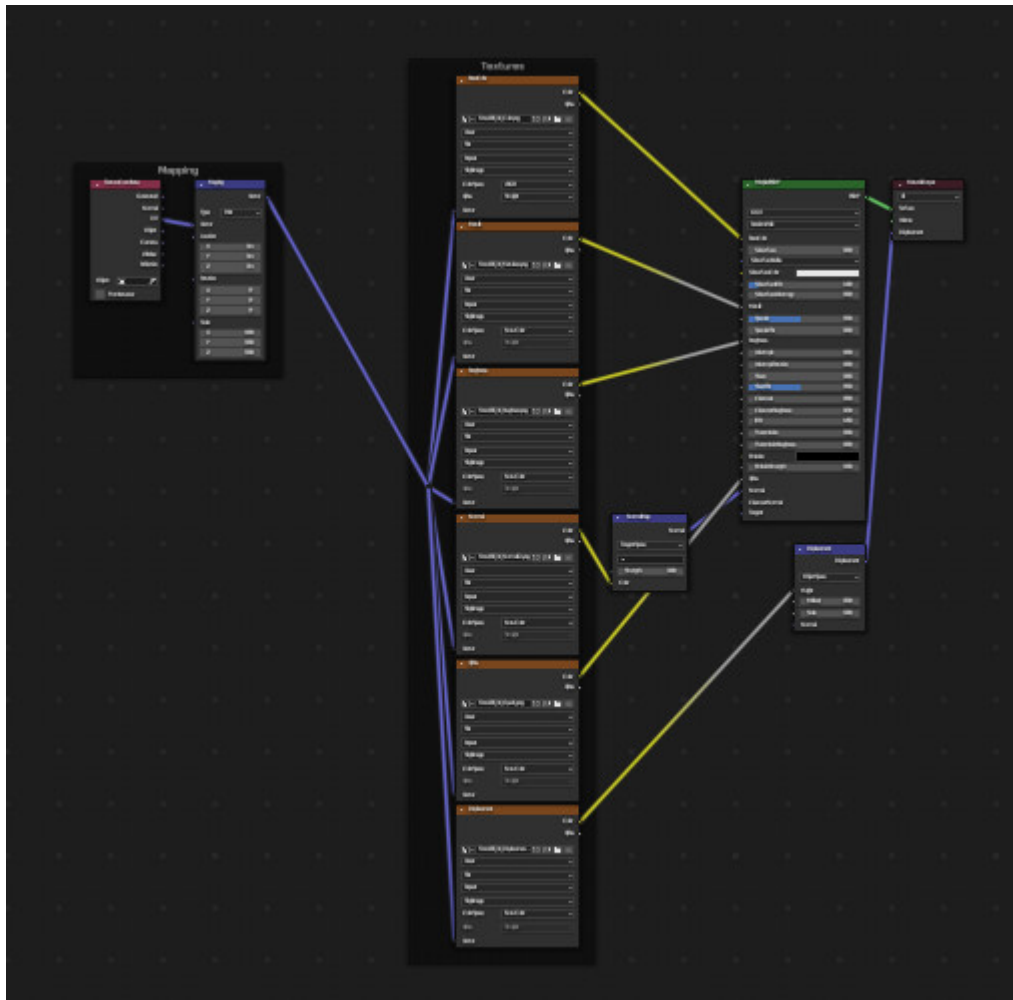
Shader

Reference

Menu:: *Node Wrangler ▶ Add Principled Setup*
Shortcut:: Shift-Ctrl-T

Add a principled texture setup from the selected texture files. Select a Principled BSDF node, select *Add Principled Setup* from the quick access menu (or press Shift-Ctrl-T), and select texture files. It automates the process of adding Image Texture nodes, loading images, selecting the appropriate Color Space, and connecting their outputs to the Principled BSDF node.

It detects the type of textures by looking at their file names. You can edit the tags used for this matching process in the add-on preferences.



Setting up these textures can take dozens of clicks, even with Node Wrangler's other tools. With Principled Texture Setup, you can reduce that to a few clicks.

Add Reroutes to Outputs

Reference

Menu:: *Node Wrangler ▶ Add Reroutes*
Shortcut:: Slash

Add reroute nodes to each output of the selected nodes.

Link Active to Selected

Reference

Menu:: *Node Wrangler ▶ Link Active to Selected*
Shortcut:: Backslash

Link the active node to the selected nodes based on various criteria.

To All Selected

Link the active node to all selected nodes. (K) You can force it to replace existing links. (Shift-K)

Use Node Name/Label

Link only to the selected nodes that have the same label as the active node. (') You can force it to replace existing links. (Shift-')

Use Outputs Names

Link only when the name of the outputs matches the name or label of the selected nodes. (;) You can force it to replace existing links. (Shift-;) This is handy for replacing sources at the same time. (For example, connecting outputs from Render Layer to image (multi-layer EXR) in Compositor.)

Align Nodes

Reference

Menu:: *Node Wrangler ▶ Align Nodes*
Shortcut:: Shift-=

Align the selected nodes horizontally or vertically. The effect is similar to scaling nodes on an axis (S X 0 or S Y 0), but it places the nodes at an even distance.

Select within Frame (Parent/Children)

-] – Select all direct child nodes of the selected frame.

- [– Select the direct parent frame node of the selected nodes.

Detach Outputs

Reference

Menu:: *Node Wrangler ▶ Detach Outputs*
Shortcut:: Shift-Alt-D

Detach the selected node's outputs while leaving linked inputs intact.

Connect to Output

Reference

Menu:: *Node Wrangler ▶ Connect to Output*
Shortcut:: 0

Connect the output of the selected node to the final output of the node tree (Composite in Compositor, Material Output or World Output in Shader, the final Group Output in Geometry Nodes, Output in Texture Nodes), or, if the node is inside a group, to the Group Output.

Add Image Sequence

Compositor Shader

Reference

Menu:: *Add ▶ Input for Compositor, or Add ▶ Texture for Shader*

Add an Image Sequence by only selecting one image from a sequence of image files. It will automatically detect the length of the sequence and set the node appropriately.

Add Multiple Images

Compositor Shader

Reference

Menu:: *Add ▶ Input for Compositor, or Add ▶ Texture for Shader*

Select multiple images and add a node for each image. (Useful for importing multiple render passes or renders for image stacking.)

See also

Please see the [old Wiki](#) for the archived original docs.

Reference

Category::	Node
Description::	Various tools to enhance and speed up node-based workflow.
Location::	<i>Node editor ▶ Sidebar</i> or see the shortcuts of individual tools.
File::	node_wrangler.py
Author::	Bartek Skorupa, Greg Zaal, Sebastian Koenig, Christian Brinkmann, Florian Meyer
License::	GPL
Note::	This add-on is bundled with Blender.

Rigging

These add-ons relate to rigging and armatures.

- [Rigify](#).

Rigify

Basics

- [Introduction](#)
 - [Main Features](#)
 - [Enabling Add-on](#)
- [Basic Usage](#)
 - [Basic Rig Generation](#)
 - [Advanced Rig Generation](#)
 - [Library Linking](#)
- [Bone Positioning Guide](#)
 - [Face Bones](#)
 - [Torso Bones](#)
 - [Limbs Bones](#)
 - [Fingers Bones](#)
- [Generated Rig Features](#)
 - [Common Features](#)
 - [Limbs](#)
 - [Fingers & Tentacles](#)
 - [Spine, Head & Tail](#)
 - [Face](#)

Customization

- [Creating Meta-rigs](#)
- [Rig Types](#)

Extensions

- [Feature Sets](#)

Development

Developer documentation is available on the [Blender Developer Documentation](#).

Reference

Category::	Rigging
Description::	Automatic rigging from building-block components.
Location::	<i>Properties ▶ Armature, Bone, 3D Viewport ▶ Tools panel, 3D Viewport ▶ Add menu ▶ Armature</i>
File::	rigify folder
Author::	Nathan Vegdahl, Lucio Rossi, Ivan Cappiello, Alexander Gavrilov
License::	GPL
Note::	This add-on is bundled with Blender.

Introduction

Rigify helps automate the creation of character rigs. It is based around a building-block approach, where you build complete rigs out of smaller rig parts (e.g. arms, legs, spines, fingers...). The rig parts are currently few in number, but as more rig parts are added to Rigify it should become more and more capable of rigging a large variety of characters and creatures.

Rigify also operates on the principle that once a rig is created, that rig should no longer need Rigify. This means you can always distribute rigs created with Rigify to people who do not have it and the rigs will still function completely.

It is important to note that Rigify only automates the creation of the rig controls and bones. It does not attach the rig to a mesh, so you still have to do skinning etc. yourself.

Main Features

Modular rigging

Rigify build blocks can be mixed together to rig any character you want. If you need to build a character with five arms and one leg, Rigify can handle it for you creating all the required complex controls system (FK, IK, and all the relative snapping tools and the UI) in few seconds.

Nondisruptive re-rig

If the generated rig doesn't fit all the features you need or, for example, you decide to add something more to your character (like a sixth arm or a tail), you can re-generate your rig without losing your previously generated features and your animation data.

Advanced and flexible feature set for character animation

The included rig samples (limbs, spines, tails, fingers, faces...) adds to all the stretchy FK/IK features a direct deformation secondary layer that